Intruder deducibility constraints with negation. Decidability and application to secured service compositions

Tigran Avanesov, Yannick Chevalier, Michael Rusinowitch, Mathieu Turuani

Intruder deducibility constraints with negation. Decidability and application to secured service compositions.

Tigran Avanesov

SnT, Université du Luxembourg, Luxembourg

Yannick Chevalier

Université Paul Sabatier & IRIT Toulouse

Michael Rusinowitch

INRIA Nancy–Grand Est & LORIA, 54600 Villers-lès-Nancy, France

Mathieu Turuani

INRIA Nancy–Grand Est & LORIA, 54600 Villers-lès-Nancy, France

Abstract

We consider a problem of automated orchestration of security-aware services under additional constraints. The problem of finding a mediator to compose secured services has been reduced in previous works to the problem of solving deducibility constraints similar to those employed for cryptographic protocol analysis. We extend in this paper the mediator synthesis procedure (i.e. a solution for the orchestration problem) by allowing additional non-disclosure policies that express the fact that some data is not accessible to the mediator at a given point of its execution. We present a decision procedure that answers the question whether a mediator satisfying these policies can be effectively synthesized. The approach presented in this work extends the constraint solving procedure for cryptographic protocol analysis in a significant way as to be able to handle negation of deducibility constraints. It applies to all subterm convergent theories and therefore covers several interesting theories in formal security analysis including encryption, hashing, signature and pairing; it is also expressive enough for some RBAC policies. A variant of this procedure for Dolev Yao theory has been implemented in Cl-Atse, a protocol analysis tool based on constraint solving.

Key words: Web services, orchestration, security policy, separation of duty, deducibility constraints, cryptographic protocols, formal methods, automated verification, synthesis

1. Introduction

1.1. Context

Trust and security management in distributed frameworks is known to be a non-trivial critical issue. It is particularly challenging in Service Oriented Architecture where services can be discovered and composed in a dynamic way. Implemented solutions should meet the seemingly antinomic goals of openness and flexibility on one hand and compliance with data privacy and other regulations on the other hand. We have demonstrated in previous works (Chevalier et al., 2008, 2012; Avanesov et al., 2012a) that functional agility can be achieved for services with a message-level security policy by providing an automated service synthesis algorithm. It resolves a system of deducibility constraints by synthesizing a *mediator* that may adapt, compose and analyze messages exchanged between client services and having the functionalities specified by a goal service. It is complete as long as the security policies only apply to the participants in the orchestration and not on the synthesized service nor on who is able to participate. However security policies often include such *non-deducibility* constraints on the mediator. For instance an organisation may not be trusted to efficiently protect the customer's data against attackers even though it is well-meaning. In this case a client would require that the mediator synthesized to interact with this organization must not have direct access to her private data, which is an effective protection even in case of total compromise. Also it is not possible to specify that the mediator enforces *e.g.* dynamic separation of duty, *i.e.*, restrictions on the possible participants based on the messages exchanged.

Since checking whether a solution computed by our previous algorithm satisfies the non-deducibility constraints is not complete, we propose in this paper to solve during the automated synthesis of the mediator both deducibility and non-deducibility constraints. The former are employed to specify a mediator that satisfies the functional requirements and the security policy on the messages exchanged by the participants whereas the latter are employed to enforce a security policy on the mediator and the participants to the orchestration.

1.1.1. Original contribution.

We have previously proposed some decision procedures (Chevalier et al., 2008, 2012; Avanesov et al., 2012a) for generating a mediator from a high-level specification with deducibility constraints of a goal service. In this paper, we extend the formalism to include non-deducibility constraints in the specification of the mediator. Then we provide a decision procedure for the resulting class of constraint systems and therefore solve the mediator synthesis problem in this setting. This paper extends the previous publication (Avanesov et al., 2012b) in several aspects: the proofs are reorganized and improved;

⋆ This work is supported by FP7 AVANTSSAR (AVANTSSAR, 2008–2010) and FP7 NESSoS (NESSoS, 2010–2014) projects.

Email addresses: tigran.avanesov@uni.lu (Tigran Avanesov), ychevali@irit.fr (Yannick Chevalier), rusi@loria.fr (Michael Rusinowitch), mathieu.turuani@loria.fr (Mathieu Turuani).

all omitted reasonings are included; the details on the implementation of the decision procedure for Dolev Yao theory within the Cl-Atse tool are given; and the experimental results for a Loan Origination Process case study with and without non-deducibility constraints are analyzed.

1.1.2. Related works.

In order to understand and anticipate potential flaws in complex composition scenarios, several approaches have been proposed for the formal specification and analysis of secure services (Armando et al., 2012; Costa et al., 2011; Armando and Ponta, 2014; Armando et al., 2013; Viganò, 2012, 2013). Among the works dedicated to trust in multi-agent systems, the models closest to ours are (Herzig et al., 2010; Lorini and Demolombe, 2008) in which one can express that an agent trusts another agent in doing or forbearing of doing an action that leads to some goal. To our knowledge no work has previously considered the automatic orchestration of security services with policies altogether as ours. However there are some interesting related attempts to analyze security protocols and trust management (Martinelli, 2005; Frau and Dashti, 2011). In (Martinelli, 2005) the author uniformly models security protocols and access control based on trust management. The work introduces an elegant approach to model automated trust negotiation. We also consider an integrated framework for protocols and policies but in our case *i*) policies can be explicitly negative such as non-disclosure policies and separation-of-duty *ii*) we propose a decision procedure for the related trust negotiation problem *iii*) we do not consider indistinguishability properties. In (Frau and Dashti, 2011) security protocols are combined with authorization logics that can be expressed with acyclic Horn clauses. The authors encode the derivation of authorization predicates (for a service) as subprotocols and can reuse in that way the constraint solving algorithm from (Millen and Shmatikov, 2001; Comon-Lundh et al., 2010) to obtain a decision procedure. In our case we consider more general intruder theories (subterm convergent ones) but focus on negation. We conjecture that our approach applies to their authorization policies too.

Our decision procedure for general (negative and positive) constraints extends (Corin et al., 2006) where negative constraints are limited to have ground terms in right-hand sides, and the deduction system is the Dolev-Yao system (Dolev and Yao, 1983), a special instance of the subterm deduction systems we consider here. In (Kähler et al., 2007) the authors study a class of contract signing protocols where some very specific Dolev-Yao negative constraints are implicitly handled (in particular only Dolev-Yao standard case is considered).

Finally one should note that the non deducibility constraints we consider tell that some data cannot be disclosed *globally* but they cannot express finer-grained privacy or information leakage notions relying on probability such as for instance differential privacy.

1.1.3. Paper organization.

In Section 2 we give motivating examples. In Subsection 2.1 we introduce a banking application and sketch our approach to obtain a mediator service. To our knowledge this application is out of the scope of alternative automatic methods.

In Section 3 we present our formal setting. A deduction system (Subsection 3.2) describes the abilities of the mediator to process the messages. The mediator synthesis problem is reduced to the resolution of constraints that are defined in Subsection 3. In Section 4 we recall the class of *subterm deduction systems* and their properties. These systems have nice properties that allow us to decide in Section 5 the satisfiability of deducibility constraints even with negation. Finally we conclude in Section 7.

Composition rules			Decomposition rules		
x, y	$\rightarrow$	$\mathrm{pair}\,(x, y)$	$\mathrm{pair}\,(x, y)$	$\rightarrow$	x
			$\mathrm{pair}\,(x, y)$	$\rightarrow$	y
x, y	$\rightarrow$	$\{\|x\|\}_y$	$y, \{\|x\|\}_y$	$\rightarrow$	x
x, y	$\rightarrow$	$\{x\}_y$	$\mathrm{inv}\,(y)\,, \{x\}_y$	$\rightarrow$	x
$x, \mathrm{inv}\,(y)$	$\rightarrow$	$\{x\}^{\mathrm{sig}}_{\mathrm{inv}(y)}$	$y, \{x\}^{\mathrm{sig}}_{\mathrm{inv}(y)}$	$\rightarrow$	x
			$x, \mathrm{rel}(x, y)$	$\rightarrow$	y
			$y, \mathrm{rel}(x, y)$	$\rightarrow$	x

Figure 1. Deduction system for the LOP example.

2. Motivating examples

2.1. Synthesis of a Loan Origination Process (LOP)

We illustrate how negative constraints are needed to express elaborated policies such as Separation of Duty by a classical loan origination process example. Our goal is to synthesize a mediator that selects two bank clerks satisfying the Separation of Duty policy to manage the client request. Such a problem is solved automatically by the decision procedure proved in the following sections. Let us walk through the specification of the different parts of the orchestration problem.

2.1.1. Formal setting.

Data are represented by first-order terms defined on a signature that comprises binary symbols for symmetric and asymmetric encryptions (resp. $\{|\_|\}_\_$, $\{\_\}_\_$), signature ($\{\_\}^{\mathrm{sig}}_\_$), and pairing (pair). Given a public key k we write $\mathrm{inv}\,(k)$ its associated private key. For example $\{a\}^{\mathrm{sig}}_{\mathrm{inv}(k)}$ is the signature of a by the owner of the pair of public and private keys $k, \mathrm{inv}\,(k)$. For readability we write $a.b.c$ a term $\mathrm{pair}\,(a, \mathrm{pair}\,(b, c))$. The construction $\mathrm{rel}(\_, \_)$ expresses that two agents are related and is used for defining a Separation of Duty policy. A unary symbol g is employed to designate the participants' identity in the "relatives" database.

A set of possible operations on data (represented by terms) is expressed in a form of so-called *deduction rules* shown in Fig. 1. These are split into two categories, composition and decomposition rules. The former are used to "compose" more complex term from simpler pieces while the latter play the inverse role of extracting a piece from a composed term. Everything that an agent can compute is deducible through this system from his initial knowledge plus the fresh data he generated (nonces) and the messages he received from external entities. As formally defined later in Section 3 this gives life to derivations, i.e. sequences of ground deduction rules augmented with nonce generations and message receptions, where any term in the left-hand side of a rule is somewhere in the right-hand sides of previous rules. A derivation shows exactly which data an agent chooses to compute (among the infinite ones) and how he do it. The deduction capabilities defined here matches the so-called Dolev-Yao model with non-atomic keys, slightly augmented with two rules for rel.

Clerk's (A) communications:

$* \Rightarrow$ A : request.M

A $\Rightarrow$ M : $g(A).\,\mathrm{pk}\,(A)$

M $\Rightarrow$ A : $\{Amnt.C.K\}_{\mathrm{pk}(A)}$

A $\Rightarrow$ M : $\{h(A.Amnt.C.Resp_A)\}^{\mathrm{sig}}_{\mathrm{inv}(pk(A))}.\{|Resp_A|\}_K$

Non-disclosure policy:

(1) M cannot deduce the fourth message before it is sent by A.
(2) M cannot deduce identity of the clerk (i.e. $g(A)$) before it is sent by A (done in the second message).

Figure 2. Clerk's communications and non-disclosure constraints

2.1.2. Client and clerks.

The client and the clerks are specified by services with a security policy, specifying the cryptographic protections and the data and security tokens, and a business logic that specifies the sequence in which the operations may be invoked. These are compiled into a sequence of protected messages each service is willing to follow, as in Fig. 2 and 3 for the clerk and the client, and following the Alice-Bob notation. Within this notation, A $\Rightarrow$ B: M is one protocol step, run after those above and before those under, where agent A sends the message M to agent B. When an agent name is unknown or irrelevant, $*$ is used instead. For consistency, an agent can only send messages that he knows or can create, and any run of the protocol will be described by a derivation showing the communications with agents in the exact same order as in each agent's specification.

Client C wants to ask for a loan from a service P, but for this he needs to get an approval from two banking clerks. He declares his intention by signing and sending to mediator M a message containing service name P and the identity of the client $g(C)$. The mediator should send back the names of two clerks A and B who will evaluate his request. The client then sends to each clerk a request containing amount $Amnt$, his name C and a fresh key N_k which should be used to encrypt decisions. Each request is encrypted with a public key of the corresponding clerk ($pk(A)$ or $pk(B)$). Then the mediator must furnish the decisions (R_a and R_b) of the two clerks, each encrypted with the proposed key N_k and accompanied by their signatures. Finally, the client uses these tokens to ask for his loan from P, where $pk(P)$ is a public key of P.

A clerk receives a request to participate in a LOP which is conducted by the mediator M. If he accepts, he returns his identity and public key. Then he receives the client's request containing data to process to evaluate the loan: the amount Amnt, the client's name C and a temporary key K to encrypt his decision. That decision is sent back together with a signature certifying its authenticity in relation to the given request.

The client's non-disclosure policy is given in Fig. 3 and is self-explanatory. Let us explain the services' non-disclosure policy. The clerk's decision (its last message) should be unforgeable, thus, it should not be known by the Mediator before it was sent by the clerk (first non-disclosure constraint of Fig. 2). The second non-disclosure constraint of Fig. 2 expresses that the clerk A can be used by the mediator only if the constraint $\natural g(A)$ is satisfied, *i.e.*, that A is not a relative with any other actor of the protocol, as either the client or the other clerk.

Client's (C) communications:

$$\begin{aligned}
&\mathrm{C} \Rightarrow \mathrm{M} : \{g(C).loan.P\}^{\mathrm{sig}}_{\mathrm{inv}(pk(C))}\\
&\mathrm{M} \Rightarrow \mathrm{C} : A.B\\
&\mathrm{C} \Rightarrow \mathrm{M} : \{Amnt.C.N_k\}_{\mathrm{pk}(A)} . \{Amnt.C.N_k\}_{\mathrm{pk}(B)}\\
&\mathrm{M} \Rightarrow \mathrm{C} : \{h(A.Amnt.C.R_a)\}^{\mathrm{sig}}_{\mathrm{inv}(pk(A))} .\{|R_a|\}_{N_k} .\\
&\qquad\qquad \{h(B.Amnt.C.R_b)\}^{\mathrm{sig}}_{\mathrm{inv}(pk(B))} .\{|R_b|\}_{N_k}\\
&\mathrm{C} \Rightarrow \mathrm{P} : \{Amnt.C.A.R_a.B.R_b\}_{\mathrm{pk}(P)} . \{h(A.Amnt.C.R_a)\}^{\mathrm{sig}}_{\mathrm{inv}(pk(A))} .\{|R_a|\}_{N_k} .\\
&\qquad\qquad\qquad\qquad\qquad\qquad \{h(B.Amnt.C.R_b)\}^{\mathrm{sig}}_{\mathrm{inv}(pk(B))} .\{|R_b|\}_{N_k}
\end{aligned}$$

Non-disclosure policy:

(1) M cannot deduce the amount $Amnt$.
(2) M cannot deduce A's decision R_a.
(3) M cannot deduce B's decision R_b.

Figure 3. Client's Communications and non-disclosure constraints

2.1.3. Goal service.

In contrast with the other services and clients, the goal service is only described in terms of possible operations and available initial data.

Initial data. Beside his private/public keys and the public keys of potential partners (*e.g.* $\mathrm{pk}(P)$) the goal service has access to a relational database denoted $\mathrm{rel}(g(a), g(c))$, $\mathrm{rel}(g(b), g(c))$, ... for storing known existing relations between agents to be checked against conflict of interests.

Deduction rules. The access to the database as well as the possible operations on messages are modeled by a set of deduction rules (formally defined later). We anticipate on the rest of this paper, and present the rules specific to this case study grouped into composition and decomposition rules in Fig. 1.

2.1.4. Mediator synthesis problem.

In order to communicate with the services (here the client, the clerks and the service P), a mediator has to satisfy a sequence of constraints expressing that *(i)* each message m expected by a service (denoted $?m$) can be deduced from all the previously sent messages m' (denoted $!m'$) and the initial knowledge, and *(ii)* each message w that should not be known or disclosed by the mediator (denoted $\natural w$ and called negative constraint) is not deducible at that point in the process.

The orchestration problem consists in finding a satisfying interleaving of the constraints imposed by each service. Since the mediator has a central position in the message exchanges, we can safely assume that all the messages goes through him. That is, each time A sends a message to B, he send it to the mediator M instead who will forward it to B. If needed the identity of B can be paired with the message to make it clear to M. One could argue that this simplification increases the knowledge of the mediator and thus, limits the solutions to the negative constraints. However, this is fine because a malicious mediator could anyway listen to and change these messages (e.g. man-in-the-middle attack), and thus, they must be counted as knowledges for the negative constraints even

if not officially going through the mediator. For instance, the clerk's and the client's constraints extracted from Fig. 2 and Fig. 3 are:

$$
\begin{cases}
Client(C) \triangleq\ !\,\{g(C).loan.P\}^{\mathrm{sig}}_{\mathrm{inv}(K_C)}\ ?A.B \\
\qquad\qquad\quad !\,\{Amnt.C.N_k\}_{\mathrm{pk}(A)}\,.\,\{Amnt.C.N_k\}_{\mathrm{pk}(B)} \\
\qquad\qquad\quad ?\,\{h(A.Amnt.C.R_a)\}^{\mathrm{sig}}_{\mathrm{inv}(pk(A))}.\{|R_a|\}_{N_k}. \\
\qquad\qquad\qquad \{h(B.Amnt.C.R_b)\}^{\mathrm{sig}}_{\mathrm{inv}(pk(B))}.\{|R_b|\}_{N_k} \\
\qquad\qquad\quad \natural Amnt\ \natural R_A\ \natural R_B \\
\qquad\qquad\quad !\,\{Amnt.C.A.R_a.B.R_b\}_{\mathrm{pk}(P)}\,.\,\{h(A.Amnt.C.R_a)\}^{\mathrm{sig}}_{\mathrm{inv}(pk(A))}.\{|R_a|\}_{N_k}. \\
\qquad\qquad\qquad\qquad\qquad\qquad \{h(B.Amnt.C.R_b)\}^{\mathrm{sig}}_{\mathrm{inv}(pk(B))}.\{|R_b|\}_{N_k} \\
Clerk(A) \triangleq\ ?request.M\ \natural g(A)\ !g(A).\,\mathrm{pk}\,(A)\ ?\,\{Amnt.C.K\}_{\mathrm{pk}(A)} \\
\qquad\qquad\quad \natural\,\{h(A.Amnt.C.Resp_A)\}^{\mathrm{sig}}_{\mathrm{inv}(pk(A))}.\{|Resp_A|\}_K \\
\qquad\qquad\quad !\,\{h(A.Amnt.C.Resp_A)\}^{\mathrm{sig}}_{\mathrm{inv}(pk(A))}.\{|Resp_A|\}_K
\end{cases}
$$

Here, each client's or clerk's list of constraints consists in a sequence of actions or assumptions presented in the order in which they must be performed or validated. For example, $?A.B$ in $Client(C)$ means that C must receive a pair of clerk's names $A.B$ from the mediator somewhere after sending his first message (declaration of intention) and before sending his second (requests to each clerk). Similarly, $\natural Amnt$ in $Client(C)$ says that C cannot continue to run in the protocol past this point if the mediator knows or can deduce $Amnt$. Each interleaving of these constraints preserving the partial orderings plus some sanity properties (origination, determination) shown in Section 3 is called a constraint system, and consists in a single sequence of constraints for the client and the clerks fused together and communicating with the mediator. A solution to a constraint system is a derivation for the mediator showing that he can validate all the constraints in the system in sequence, i.e. with the same ordering he *i)* can create any message t expected by an agent ($?t$) from his initial knowledge plus the nonces he generated (fresh data) and the messages sent to him earlier in the derivation; *ii)* cannot create any message t in a negative constraint ($\natural t$) from his knowledge plus the nonces he generated and the messages sent to him earlier in the derivation. Since any constraint system has the same length as the orchestration problem from which it is derived, and since we target an NP decision procedure for building a solution (derivation) to that problem, we assume for simplicity in the rest of this paper that a single ordering has been chosen already, and thus, that only one constraint system needs to be checked. The result can then be trivially lifted to finding a solution for an orchestration problem by guessing together one constraint system (same length) and one derivation (bounded by some fixed and *known* polynom), and then checking at the same time if the constraint system is a satisfiable ordering of the orchestration problem and if the derivation is indeed a solution to it. A solution produced by our procedure can then be translated automatically into a mediator (the derivation is complete). An example of this is shown in Section 6.3. Note, for example, that without the negative constraint $\natural g(A)$ a synthesized mediator might accept any clerk identity and that could violate the Separation of Duty policy.

2.2. Expressing Role Based Access Control mechanisms

We show now how our service synthesis method can accommodate constraints derived from Role Based Access Control policies (Ferraiolo and Kuhn, 1992). Role Based Access Control (RBAC in short) has been introduced in organizations to simplify the management of individual user rights. Users can be assigned some *roles*, and through these roles obtain permissions to perform various operations. For illustration purpose we present below some RBAC policies and their encoding.

Let us first give some notations:

- atomic permission: $a_1(X), \ldots, a_k(X)$, where a_i represents a permission and X is an agent, i.e. $a_1(b)$ stands for "agent b has a permission a_1" [1];
- role: A role is represented by a function symbol that, for the purpose of RBAC modelling, is applied on the set of atomic permissions and a set of inherited roles, also modelled by similar terms;
- role hierarchy: Roles can be hierarchically organized. Higher-level roles inherit permissions owned by sub-roles. For example, a "project leader" role subsumes a "programmer" role, since all atomic permissions of the "programmer" role are inherited by the "project leader" role.

Assume that a "project leader" role is assigned to some agent X. This will be represented by the following term where the root symbol is the highest role of X in the hierarchy and its arguments list contains the specific permissions given to a"project leader" (e.g. $can\_fire\_programmer$) and the sub-roles inherited from the hierarchy (here $programmer$):

$$R_{leader}(can\_fire\_programmer(X), R_{programmer}(can\_commit(X)))$$

We add to the current deduction rules of Figure 1 some extra rules $R_{leader}(X_1, X_2) \rightarrow X_i$, for $i \in \{1, 2\}$ and $R_{programmer}(Y) \rightarrow Y$. These rules encode the possibility to obtain all available atomic permissions from a role, as well as all permissions from sub-roles of the hierarchy.

To be able to assign a project leader role to an agent, the following agent's actions may be used:

$$?X!R_{leader}(can\_fire\_programmer(X), can\_commit(X), R_{programmer}(can\_commit(X)))$$

2.2.1. Delegation of rights

A simple pair of receive-send actions may be used to specify a delegation, where Agent X delegates to Agent Y a right a_1 that X has:

$$?X.Y.a_1(X)\ !deleg(X, Y, a_1(X), a_1(Y))$$

An extra deduction rule is required too: $deleg(X, Y, Z, T) \rightarrow T$ [2]. Once the right is delegated, this rule permits to infer that right.

Note that a simplified encoding of the right delegation property $?X.Y.a_1(X)\ !a_1(Y)$ that avoids using *deleg* symbol does not permit one to forbid a delegation at some point using the $\natural$ policy; however with *deleg* symbol it is possible: $\natural deleg(X, Y, a_1(X), a_1(Y))$ ensures that the agent X has not delegated right a_1 to agent Y (see also § 2.2.4).

---

[1] The notation can be extended to take into account *objects* in order to express policies like "agent b has permission a_1 on object c".

[2] We might also add X, Y, Z to the right hand side, but these values are supposed to be known.

2.2.2. Separation of roles

This policy specifies that an agent cannot play two specific roles. To express such a policy in a service composition we simply append at the end:

$$\natural(\bar{R}_{clerk}(X).\bar{R}_{client}(X))$$

That is, $\bar{R}_{clerk}(X)$ and $\bar{R}_{client}(X)$ are not known at the same time, meaning that X does not play clerk and client roles in parallel.

2.2.3. Separation of duties

This policy specifies that an agent cannot have two specific permissions at the same time. To express such a policy we simply add:

$$\natural(a(X).b(X))$$

2.2.4. Delegation restriction

An agent A can be forbidden to delegate some right a. To express such a policy in a service composition we simply add:

$$\natural(a(A).deleg(A, X, a(A), a(X)))$$

3. Derivations and constraint systems

In our setting, messages are terms generated or obtained according to some elementary rules called *deduction rules*. A *derivation* is a sequence of deduction rules applied by a mediator to build new messages. The goal of the synthesis is specified by a *constraint system*, *i.e.* a sequence of terms labelled by symbols !,? or $\natural$, respectively *sent*, *received*, or *unknown* at some step of the process.

3.1. Terms and substitutions

Let $\mathcal{X}$ be a set of *variables*, $\mathcal{F}$ be a set of *function symbols* and $\mathcal{C}$ be a set of *constants*. The set of *terms* $\mathcal{T}$ is the minimal set containing $\mathcal{X}$, $\mathcal{C}$ and if $t_1, \ldots, t_k \in \mathcal{T}$ then $f(t_1, \ldots, t_k) \in \mathcal{T}$ for any $f \in \mathcal{F}$ with arity k. The set of *subterms* of a term t is denoted $\mathrm{Sub}(t)$ and is the minimal set containing t such that $f(t_1, \ldots, t_n) \in \mathrm{Sub}(t)$ implies $t_1, \ldots, t_n \in \mathrm{Sub}(t)$ for $f \in \mathcal{F}$. We denote $\mathrm{Vars}\,(t)$ the set $\mathcal{X} \cap \mathrm{Sub}(t)$. A term t is *ground* is $\mathrm{Vars}\,(t) = \emptyset$. We denote $\mathcal{T}_g$ the set of ground terms.

A *substitution* σ is an idempotent mapping from $\mathcal{X}$ to $\mathcal{T}$. The application of a substitution σ on a term t is denoted $t\sigma$ and is equal to the term t where each variable x has been replaced by the term $x\sigma$. The *domain* of σ (denoted by $\mathrm{dom}\,(\sigma)$) is set: $\{x \in \mathcal{X} : x\sigma \neq x\}$. The *image* of σ is $\mathrm{img}\,(\sigma) = \{x\sigma : x \in \mathrm{dom}\,(\sigma)\}$. A substitution σ is ground if $\mathrm{img}\,(\sigma) \subseteq \mathcal{T}_g$. We say that a substitution σ is *injective* on a set of terms T, iff for all $p, q \in T$ $p\sigma = q\sigma$ implies $p = q$. Given two substitutions σ, δ, the substitution $\sigma\delta$ has for domain $\mathrm{dom}\,(\sigma) \cup \mathrm{dom}\,(\delta)$ and is defined by $x\sigma\delta = (x\sigma)\delta$. If $\mathrm{dom}\,(\sigma) \cap \mathrm{dom}\,(\delta) = \emptyset$ we write $\sigma \cup \delta$ instead of $\sigma\delta$.

A *unification system* U is a finite set of equations $\{p_i =_? q_i\}_{1 \leq i \leq n}$ where $p_i, q_i \in \mathcal{T}$. A substitution σ is a *unifier* of U or equivalently satisfies U iff for all $i = 1, \ldots, n$, $p_i\sigma = q_i\sigma$. Any satisfiable unification system U admits a *most general unifier* $\mathrm{mgu}\,(U)$, unique modulo variable renaming, and such that for any unifier σ of U there exists a substitution τ such that $\sigma = \mathrm{mgu}\,(U)\,\tau$. We assume in the rest of this paper that

$\mathrm{Vars}\,(\mathrm{img}\,(\mathrm{mgu}\,(U))) \subseteq \mathrm{Vars}\,(U)$, *i.e.*, the most general unifier does not introduce new variables.

A *sequence* s is indexed by $[1,\ldots,n]$ with $n \in \mathbb{N}$. We write $|s|$ the length of s, $\emptyset$ the empty sequence, $s[i]$ the i-th element of s, $s[m:n]$ the sequence $s[m],\ldots,s[n]$ and s,s' the concatenation of two sequences s and s'. We write $e \in s$ and $E \subseteq s$ instead of, respectively, there exists i such that $s[i] = e$ and for all $e \in E$ we have $e \in s$.

While not explicited, the size of any object is assumed to be linear (with some fixed coef.) in the DAG-size of the set of it's elements.

3.2. Deduction systems

The new values created by the mediator are constants in a subset $\mathcal{C}_{\mathrm{med}}$ of $\mathcal{C}$. We assume that both $\mathcal{C}_{\mathrm{med}}$ and $\mathcal{C} \setminus \mathcal{C}_{\mathrm{med}}$ are infinite. Given $l_1,\ldots,l_n,r \in \mathcal{T}$, the notation $l_1,\ldots,l_n \to r$ denotes a *deduction rule* if $\mathrm{Var}(r) \subseteq \bigcup_{i=1}^{n} \mathrm{Var}(l_i)$. A *deduction* is a ground instance of a deduction rule. A *deduction system* is a set of deduction rules that contains a finite set of deduction rules in addition to all *nonce creation rules* $\to n$ (one for every $n \in \mathcal{C}_{\mathrm{med}}$) and all reception rules $?t$ (one for every $t \in \mathcal{T}$). All rules but the reception rules are called *standard* rules. The deduction system describes the abilities of the mediator to process the messages. In the rest of this section we fix an arbitrary deduction system $\mathcal{D}$. We denote by $l \twoheadrightarrow r$ any rule and $l \to r$ any standard rule.

3.3. Derivations and localizations

A *derivation* is a sequence of deductions, including receptions of messages from available services, performed by the mediator. Given a sequence of deductions $E = (l_i \twoheadrightarrow r_i)_{i=1,\ldots,m}$ we denote $\mathrm{R}_E\,(i)$ the set $\{r_j : j \leq i\}$.

Definition 3.1 (Derivation)**.** A sequence of deductions $D = (l_i \twoheadrightarrow r_i)_{i=1,\ldots,m}$ is a *derivation* if for any $i \in \{1,\ldots,m\}$, $l_i \subseteq \mathrm{R}_D\,(i-1)$.

Given a derivation D we define $\mathrm{Next}_D(i) = \min(\{|D|+1\} \cup \{j : j > i \text{ and } D[j] = ?t_j\})$. The explicit knowledge of the mediator is the set of terms it has already deduced, and its implicit knowledge is the set of terms it can deduce. If the former is K we denote the latter $\mathrm{Der}(K)$. A derivation D is a *proof* of $t \in \mathrm{Der}(K)$ if $?r \in D$ implies $r \in K$, and $D[|D|] = l \twoheadrightarrow t$. Thus, we have:

$$\mathrm{Der}(K) = \{t : \exists D \text{ derivation s.t. } ?r \in D \text{ implies } r \in K, \text{ and } D[|D|] = l \to t\}$$

3.4. Constraint systems

Definition 3.2 (Constraint system)**.** A constraint system $\mathcal{S}$ is a sequence of constraints where each constraint has one of three forms (where t is a term):

(1) $?t$, denoting a message reception by an available service or a client,
(2) $!t$, denoting a message emission by an available service or a client,
(3) $\natural t$, a negative constraint, denoting that the mediator must not be able to deduce t at this point;

and that satisfies the following properties for any $1 \leq i \leq |\mathcal{S}|$:

Origination: if $\mathcal{S}[i] = !t_i$ then $\mathrm{Vars}\,(t_i) \subseteq \bigcup_{j<i} \mathrm{Vars}\,(\{t_j : \mathcal{S}[j] = ?t_j\})$;
Determination: if $\mathcal{S}[i] = \natural t_i$ then $\mathrm{Vars}\,(t_i) \subseteq \bigcup_j \mathrm{Vars}\,(\{t_j : \mathcal{S}[j] = ?t_j\})$.

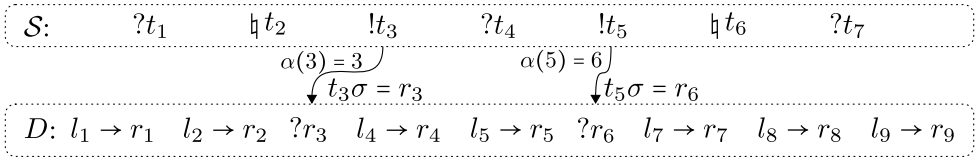

Figure 4. A constraint system and a compliant derivation

Origination means that every unknown in a service's state originates from previous input by the mediator. *Determination* means that negative constraints are on messages determined by a service's state at the end of its execution.

In the rest of this paper, $\mathcal{S}$ (and decorations thereof) denotes a constraint system. An index i is a *send* (*resp.* a *receive*) index if $\mathcal{S}[i] = !t$ (*resp.* $\mathcal{S}[i] = ?t$) for some term t. If $i_1, \ldots, i_k$ is the sequence of all send (*resp.* receive) indices in $\mathcal{S}$ we denote $\mathrm{Out}(\mathcal{S})$ (*resp.* $\mathrm{In}(\mathcal{S})$) the sequence $\mathcal{S}[i_1], \ldots, \mathcal{S}[i_k]$. We note that the origination and determination properties imply $\mathrm{Var}(\mathcal{S}) = \mathrm{Var}(\mathrm{In}(\mathcal{S}))$. Given $1 \leq i \leq |\mathcal{S}|$ we denote $\mathrm{prev}_{\mathcal{S}}(i)$ to be $\max(\{0\} \cup \{j : j \leq i \text{ and } \mathcal{S}[j] = !t_j\})$.

Definition 3.3 (Solution of a constraint system)**.** A ground substitution σ is a solution of $\mathcal{S}$, and we denote $\sigma \models \mathcal{S}$, if $\mathrm{dom}(\sigma) = \mathrm{Var}(\mathcal{S})$ and
(1) if $\mathcal{S}[i] = ?t$ then $t\sigma \in \mathrm{Der}(\{t_j\sigma : j \leq \mathrm{prev}_{\mathcal{S}}(i) \text{ and } \mathcal{S}[j] = !t_j\})$
(2) if $\mathcal{S}[i] = \natural t$ then $t\sigma \notin \mathrm{Der}(\{t_j\sigma : j \leq \mathrm{prev}_{\mathcal{S}}(i) \text{ and } \mathcal{S}[j] = !t_j\})$

Definition 3.4 (Compliant derivations)**.** Let σ be a ground substitution with $\mathrm{dom}(\sigma) = \mathrm{Var}(\mathcal{S})$. A derivation D is $(\mathcal{S}, \sigma)$-*compliant* if there exists a strictly increasing bijective mapping α from the send indices of $\mathcal{S}$ to the set $\{j : D[j] = ?r\}$ such that $\mathcal{S}[i] = !t$ implies $D[\alpha(i)] = ?t\sigma$.

An example of $(\mathcal{S}, \sigma)$-compliant derivation is shown in Figure 4. Since a sequence of receptions is a derivation, we note that for every ground substitution σ with $\mathrm{dom}(\sigma) = \mathrm{Var}(\mathrm{In}(\mathcal{S}))$ there exists at least one compliant derivation D.

Definition 3.5 (Proof of a solution)**.** Let σ be a ground substitution. A derivation D is a *proof* of $\sigma \models \mathcal{S}$, and we denote $D, \sigma, \alpha \vdash \mathcal{S}$, if:
(1) D is $(\mathcal{S}, \sigma)$-compliant with the mapping α and
(2) if $\mathcal{S}[i] = ?t$ there is $j < \mathrm{Next}_D(\alpha(\mathrm{prev}_{\mathcal{S}}(i)))$ such that $D[i] = l \twoheadrightarrow t\sigma$ and
(3) if $\mathcal{S}[i] = \natural t$ then $t\sigma \notin \mathrm{Der}(\{t_j\sigma : j \leq \mathrm{prev}_{\mathcal{S}}(i) \text{ and } \mathcal{S}[j] = !t_j\})$.

In Figure 4, if σ is a solution of $\mathcal{S}$ and, for example, $t_1\sigma = r_2$, $t_2\sigma \notin \mathrm{Der}(\emptyset)$, $t_4\sigma = r_4$, $t_6\sigma \notin \mathrm{Der}(\{r_3, r_6\})$ and $t_7\sigma = r_8$ then D is a proof of $\sigma \models \mathcal{S}$.

A technical subtility induced by the negative constraints is that it is easier to work on derivations that are not necessarily solutions of the constraint system and prove that if the constraint system is satisfiable one of these derivations is among its proofs. Accordingly, we define *maximal* derivations with regard to a set of terms T and a substitution σ as a derivation in which every subterm of T whose σ-instance is deducible is deduced as soon as possible (Def. 3.6). Then we prove that if a derivation D is (T, σ)-maximal and (T', σ') extends in a natural way (T, σ) then D can be extended into a (T', σ')-maximal derivation (Lemma 2). This lemma can be applied on any $(\mathcal{S}, \sigma)$ compliant derivation to obtain a $(\mathcal{S}, \sigma)$ compliant $(\mathrm{Sub}(\mathcal{S}), \sigma)$ maximal derivation. Such derivations are proofs of $\sigma \models \mathcal{S}$ if σ satisfies $\mathcal{S}$ (Lemma 1).

Definition 3.6 (Maximal derivation)**.** Let T be a finite set of terms and σ be a ground substitution with $\mathrm{dom}(\sigma) = \mathrm{Var}(T)$. A derivation D is (T,σ)-maximal if for every $t \in \mathrm{Sub}(T)$, $t\sigma \in \mathrm{Der}(\mathrm{R}_D(i))$ iff $t\sigma \in \mathrm{R}_D(\mathrm{Next}_D(i)-1)$.

First we prove that maximal derivations are natural proof candidates of $\sigma \models \mathcal{S}$ by showing that whether an individual constraint is satisfied by a substitution σ can be read on a maximal compliant derivation.

Lemma 1. Let σ be a ground substitution with $\mathrm{dom}(\sigma) = \mathrm{Var}(\mathcal{S})$ and D be a $(\mathcal{S},\sigma)$-compliant $(\mathrm{Sub}(\mathcal{S}),\sigma)$-maximal derivation. **Then** $\sigma \models \mathcal{S}$ iff for all i

- if $\mathcal{S}[i] = ?t$ then there exists $j < \mathrm{Next}_D(\alpha(\mathrm{prev}_{\mathcal{S}}(i))) : D[j] = l \twoheadrightarrow t\sigma$ and
- if $\mathcal{S}[i] = \natural t$ then for all $j < \mathrm{Next}_D(\alpha(\mathrm{prev}_{\mathcal{S}}(i))) : D[j] \neq l \twoheadrightarrow t\sigma$.

In the next lemma we show that any (T,σ)-maximal derivation D may be extended into a (T',σ')-maximal derivation for an arbitrary extension T',σ' of T,σ by adding into D only standard deductions.

Lemma 2. Let σ be a ground substitution with $\mathrm{dom}(\sigma) = \mathrm{Var}(\mathcal{S})$. Let T_1, T_2 be two sets of terms such that $T_1 \subseteq T_2$, and σ_1, σ_2 be two substitutions such that $\mathrm{dom}(\sigma_1) = \mathrm{Var}(T_1)$ and $\mathrm{dom}(\sigma_2) = \mathrm{Var}(T_2) \setminus \mathrm{Var}(T_1)$. If D is a (T_1,σ_1)-maximal $(\mathcal{S},\sigma)$-compliant derivation in which no term is deduced twice by a standard rule, **then** there exists a $(T_2, \sigma_1 \cup \sigma_2)$-maximal $(\mathcal{S},\sigma)$-compliant derivation D' in which no term is deduced twice by a standard rule such that every deduction whose right-hand side is in $\mathrm{Sub}(T_1)\sigma_1$ occurs in D' iff it occurs in D.

Proof. Let $i_1,\ldots,i_k$ be the indices of the non-standard rules in D, let $D[i_j] = ?t_{i_j}$, and let for $0 \leq j \leq k$ $D_j = D[i_j + 1 : i_{j+1} - 1]$ with $i_0 = 0$ and $i_{k+1} = |D| + 1$. That is, $D = D_0, ?t_{i_1}, D_1, ?t_{i_2}, D_2, \ldots ?t_{i_k}, D_k$. Noting that $\mathrm{dom}(\sigma_1) \cap \mathrm{dom}(\sigma_2) = \emptyset$ let $\sigma' = \sigma_1 \cup \sigma_2$.

For each $t \in \mathrm{Sub}(T_2)$ such that $t\sigma' \in \mathrm{Der}(t_{i_1},\ldots,t_{i_k})$ let i_t be minimal such that $t\sigma' \in \mathrm{Der}(t_{i_1},\ldots,t_{i_t})$, and let E^0_t be a proof of this fact, and E_t be a sequence of standard deductions obtained by removing every non-standard deduction from E^0_t.

For $0 \leq j \leq k$ let D'_j be the sequence of standard deduction steps $D_j, E_{s_1},\ldots,E_{s_p}$ for all $s_m \in \mathrm{Sub}(T_2)\sigma' \setminus \mathrm{Sub}(T_1)\sigma'$ such that $i_{s_m} = j$ in which every rule of $E_{s_1},\ldots,E_{s_p}$ that deduces a term previously deduced in the sequence or for some $m \leq j$ deduced in D'_m or in $D[i_m]$ is removed.

Let $D' = D'_0, ?t_{i_1}, D'_1, \ldots, ?t_{i_k}, D'_k$. We have deleted in each E^0_t only deductions whose right-hand side occurs before in D', and thus D' is a derivation. Since the D'_i contains only standard deductions, we can see that D' is $(\mathcal{S},\sigma)$-compliant.

Since D is (T_1,σ_1)-maximal and no term is deduced twice in D we note that, for $t \in T_1$, no standard deduction of $t\sigma_1$ from a sequence D_j is deleted. Furthermore we note that standard deductions of terms in $\mathrm{Sub}(T_2)\sigma_2$ that are also in $\mathrm{Sub}(T_1)\sigma_1$ are deleted by construction and by the maximality of D. Thus a deduction whose right-hand side is in $\mathrm{Sub}(T_1)\sigma_1$ is in D' iff it occurs in D.

By construction D' is (T_2,σ')-maximal and no term is deduced twice by standard deductions. □

Informally, the condition on the deductions whose right-hand side is in $\mathrm{Sub}(T_1)\sigma$ enforces that the constructed derivation D' is an extension of D with new deductions.

Taking $T_1 = \emptyset$, $T_2 = \mathrm{Sub}(\mathcal{S})$, and $\sigma_2 = \sigma$, Lemma 2 implies that for every substitution σ of domain $\mathrm{Var}(\mathcal{S})$ there exists a $(\mathcal{S}, \sigma)$-compliant $(\mathrm{Sub}(\mathcal{S}), \sigma)$-maximal derivation D. By Lemma 1 if $\sigma \models \mathcal{S}$ then D is a proof of $\sigma \models \mathcal{S}$. Since the converse is trivial, it suffices to search proofs maximal with regard to $T \supseteq \mathrm{Sub}(\mathcal{S})$.

Lemma 3. If $\sigma \models \mathcal{S}$ then there exists a $(\mathcal{S}, \sigma)$-compliant and $(\mathrm{Sub}(\mathcal{S}), \sigma)$-maximal derivation D and α such that $D, \sigma, \alpha \vdash \mathcal{S}$.

4. Subterm deduction system

In this section we will prove some properties on derivations allowing us to show bounds on sets of terms from which the derivations' messages are instantiated (Lemma 5 and 8). Moreover, we also introduce the notion of milestone sequences Definition 4.3, which are maximal sequences of constraints matching a derivation with "good" subterm properties. The purpose is to identify in Section 5 particular solutions to constraint systems which sizes are bounded by a fixed polynom that we can compute.

4.1. Definition and main property

We say that a deduction system is a *subterm deduction system* whenever each deduction rule which is not a nonce creation or a message reception is either:

(1) $x_1, \ldots, x_n \to f(x_1, \ldots, x_n)$ for a function symbol f;

(2) $l_1, \ldots, l_n \to r$ for some terms $l_1, \ldots, l_n, r$ such that $r \in \bigcup_{i=1}^{n} \mathrm{Sub}(l_i)$.

A *composition* rule is either a message reception, a nonce creation, or a rule of the first type. A deduction rule is otherwise a *decomposition* rule. In the rest of this paper we will usually also write (de)composition rule to denote a ground instance of one such rule. Reachability problems for deduction systems with a convergent equational theory are reducible to the satisfiability of a constraint system in the empty theory for a deduction system in our setting (Lynch and Meadows, 2005; Kourjieh, 2009). If furthermore the equational theory is *subterm* (Baudet, 2005) the reduction is to a subterm deduction system as just defined above.

Now we show that if $D, \sigma, \alpha \vdash \mathcal{S}$, a term $s \in \mathrm{Sub}(D)$ is either the instance of a non-variable subterm of $\mathrm{Out}(\mathcal{S})$ or deduced by a standard composition.

Lemma 4. Let σ be a ground substitution such that $\sigma \models \mathcal{S}$. If D is a proof of $\sigma \models \mathcal{S}$ such that no term is deduced twice in D by standard rules and s is a term such that $s \in \mathrm{Sub}(D)$ and $s \notin (\mathrm{Sub}(\mathrm{Out}(\mathcal{S})) \setminus \mathcal{X})\sigma$ **then** there exists an index i in D such that $D[i] = l \to s$ is a composition rule and $s \notin \mathrm{Sub}(\mathrm{R}_D\,(i-1))$.

Proof. First we note that by definition of subterm deduction systems for any decomposition rule $l \to r$ we have *a)* $r \in \mathrm{Sub}(l)$, and *b)* for any composition rule $l \to r$ we have $l \subset \mathrm{Sub}(r)$ and $\mathrm{Sub}(r) \setminus \mathrm{Sub}(l) = \{r\}$.

Let D be a proof of $\sigma \models \mathcal{S}$, and let i be minimal such that $D[i] = l_r \twoheadrightarrow r$ with $s \in \mathrm{Sub}(r)$. Since $l_r \subseteq \mathrm{R}_D\,(i-1)$, the minimality of i implies $s \in \mathrm{Sub}(r) \setminus \mathrm{Sub}(l_r)$.

Thus by *a)* $D[i]$ cannot be a decomposition.

If $D[i] = ?r$ then by the $(\mathcal{S}, \sigma)$-compliance of D we have $\mathcal{S}[\alpha^{-1}(i)] = !t$ with $t\sigma = r$ and $t \in \mathrm{Sub}(\mathrm{Out}(S))$. We have $s \in \mathrm{Sub}(r) = \mathrm{Sub}(t\sigma) = (\mathrm{Sub}(t) \setminus \mathrm{Vars}\,(t))\sigma \cup \mathrm{Sub}(\mathrm{Vars}\,(t)\,\sigma)$.

If $s \in (\mathrm{Sub}(\mathrm{Out}(S)) \setminus \mathcal{X})\sigma$ we are done, otherwise there exists $y \in \mathrm{Vars}(t)$ with $s \in \mathrm{Sub}(y\sigma)$. By the origination property, there exists $k < \alpha^{-1}(i)$ such that $\mathcal{S}[k] =? t'$ with $y \in \mathrm{Vars}(t')$. Since $D, \sigma, \alpha \vdash \mathcal{S}$ and $k < \alpha^{-1}(i)$ there exists $j < i$ such that $D[j] = l_j \to t'\sigma$. The minimality of i is contradicted by $s \in \mathrm{Sub}(t'\sigma)$.

Therefore, $D[i] = l_r \to r$ is an instance of a standard composition rule. As a consequence, $\mathrm{Sub}(r) \setminus \mathrm{Sub}(l_r) = \{r\}$. Since $s \in \mathrm{Sub}(r) \setminus \mathrm{Sub}(l_r)$, we finally obtain $s = r$. □

4.2. Locality

Subterm deduction systems are not necessarily local in the sense of (McAllester, 1993). However we prove in this subsection that given σ, there exists a finite extension T of $\mathrm{Sub}(\mathcal{S})$ and an extension σ' of σ of domain $\mathrm{Var}(T)$ and a (T, σ')-maximal derivation D in which every deduction relevant to the proof of $\sigma \models \mathcal{S}$ is liftable into a deduction between terms in T. Let us first clarify the above statements.

Definition 4.1 (Localization set)**.** A set of terms T *localizes* a derivation $D = (l_i \twoheadrightarrow r_i)_{1 \le i \le m}$ for a substitution σ of domain $\mathrm{Var}(T)$ if for every $1 \le i \le m$ if $D[i]$ is a standard rule and there exists $t \in \mathrm{Sub}(T) \setminus \mathcal{X}$ such that $t\sigma = r_i$, there exists $t_1, \ldots, t_n \in \mathrm{Sub}(T)$ such that $\{t_1\sigma, \ldots, t_n\sigma\} \subseteq \mathrm{R}_D(i-1)$ and $t_1, \ldots, t_n \to t$ is the instance of a standard deduction rule.

First, we prove that for subterm deduction systems, every proof D of $\sigma \models \mathcal{S}$ is localized by a set T of DAG size linear in the DAG size of $\mathcal{S}$. Note that the coefficient for this linearity is bounded by the size of the deduction system.

Lemma 5. If σ is a ground substitution such that $\sigma \models \mathcal{S}$ there exists $T \supseteq \mathrm{Sub}(\mathcal{S})$ of size bounded by $v \times |\mathrm{Sub}(\mathcal{S})|$ (with v the size of the deduction system), a substitution τ of domain $\mathrm{Var}(T) \setminus \mathrm{Var}(\mathcal{S})$ and a $(T, \sigma \cup \tau)$-maximal and $(\mathcal{S}, \sigma)$-compliant derivation localized by T for $\sigma \cup \tau$.

Proof. By Lemma 2 applied with $T_1 = \emptyset$, $T_2 = \mathrm{Sub}(\mathcal{S})$, $\sigma_1 = \emptyset, \sigma_2 = \sigma$, and D_0 the $(\mathcal{S}, \sigma)$-compliant derivation that has no standard deductions, there exists a $(\mathrm{Sub}(\mathcal{S}), \sigma)$-maximal $(\mathcal{S}, \sigma)$-compliant derivation D in which no term is deduced twice by a standard deduction. From now on we let $T_0 = \mathrm{Sub}(\mathcal{S})$.

Let $\{l_i \to r_i\}_{1 \le i \le n}$ be the set of decompositions in D, and $\{(L_i \to R_i, \tau_i)\}_{1 \le i \le n}$ be a set of decomposition rules and ground substitutions such that for all $1 \le i \le n$ we have $L_i\tau_i \to R_i\tau_i = l_i \to r_i$. Since no term in D is deduced twice by a standard deduction, by Lemma 4 we have $n \le |\mathrm{Sub}(\mathrm{Out}(S))|$.

Modulo variable renaming we may assume that $i \neq j$ implies $\mathrm{dom}(\tau_i) \cap \mathrm{dom}(\tau_j) = \emptyset$, and thus that $\tau = \bigcup_{i=1}^{n} \tau_i$ is defined on $T_1 = \bigcup_{i=1}^{n} (\mathrm{Sub}(L_i) \cup \mathrm{Sub}(R_i))$. Note that the size of T_1 is bounded by $M \times |\mathrm{Sub}(\mathrm{Out}(\mathcal{S}))|$, where M is the maximal size of a decomposition rule belonging to the deduction system.

Let $T = T_0 \cup T_1$ and, noting that these substitutions are defined on non-intersecting domains, let $\sigma' = \sigma \cup \tau$. By construction the size of T is bounded by $(M+1) \times |\mathrm{Sub}(\mathcal{S})|$, and $(M+1)$ by the size of the deduction system (it has more than one rule).

By Lemma 2 there exists a $(\mathcal{S}, \sigma)$-compliant derivation D' which is (T, σ')-maximal and such that each deduction of a term in $T_0\sigma$ that occurs in D also occurs in D' and no term is deduced twice in D' by a standard deduction.

Let $l \to r$ be a deduction in D' which does not appear in D. Since D is (T_0, σ)-maximal we have $r \notin \mathrm{Sub}(T_0)\sigma$, and thus $r \notin \mathrm{Sub}(\mathrm{Out}(\mathcal{S}))\sigma$. Since no term is deduced twice in D' by Lemma 4 this deduction must be a composition.

Let us prove D' is (T, σ')-localized. By definition of composition rules, every composition that deduces a term $t\sigma'$ with $t \in \mathrm{Sub}(T) \setminus \mathrm{Var}(T)$ has a left-hand side $t_1\sigma', \ldots, t_k\sigma'$ with $t_1, \ldots, t_k \in \mathrm{Sub}(T)$ and $t_1, \ldots, t_k \to t$ is an instance of a composition rule. By the preceding paragraph every decomposition in D' occurs in D and thus by construction has its left-hand side in $T_1\sigma'$ which was previously built in D and is an instance of some $L_i \to R_i$ such that $\mathrm{Sub}(L_i \cup \{R_i\}) \subseteq T_1 \subseteq T$.

Thus every deduction whose right-hand side is in $(\mathrm{Sub}(T) \setminus \mathrm{Var}(T))\sigma'$ has its left-hand side in $\mathrm{Sub}(T)\sigma'$, and thus D' is localized by T for σ'. □

4.3. One-to-one localization

We prove now that to solve constraint systems one can first guess equalities between terms in T and then solve constraint systems without variables. The guess of equalities is correct with regard to a solution σ if terms in T that have the same instance by σ are syntactically equal. We characterize these guesses as follows.

Definition 4.2 (One-to-one localizations)**.** A set of terms T *one-to-one localizes* a derivation D for a ground substitution σ if σ is injective on $\mathrm{Sub}(T)$ and T localizes D for σ.

In the following sequence of Lemmas, we first prove in 6 and 7 routine properties of most general unifiers. They imply that from any unifier σ of a unification system $\mathcal{U}$ one can build a most general unifier θ such that $\theta\sigma = \sigma$ and more importantly $\mathrm{Sub}(\mathcal{U})\theta = \mathrm{Sub}(\mathcal{U}\theta)$. This property is employed in Lemma 8 to prove that once equalities between subterms are correctly guessed there exists a one-to-one localization of a maximal proof D whose size is linear (polynomial would have sufficed for our purpose) in the size of the input constraint system.

Lemma 6. Let T be a set of terms such that $T = \mathrm{Sub}(T)$, σ be a ground substitution defined on $\mathrm{Vars}(T)$, $U = \{p =_? q : p, q \in T \wedge p\sigma = q\sigma\}$ be a unification system and θ be its most general idempotent unifier with $\mathrm{Vars}(\mathrm{img}(\theta)) \subseteq \mathrm{Vars}(U)$. Then for any term t, $t\theta\sigma = t\sigma$.

Proof. Let us show that $\forall x \in \mathrm{Vars}(T), x\sigma = x\theta\sigma$. Note that this trivially holds if $x\theta = x$, and thus we consider the case $x\theta \neq x$. Since U contains all equations $p =_? p$ for $p \in \mathrm{Sub}(T) = T$, we have $\mathrm{Sub}(T) = \mathrm{Sub}(U)$. From the idempotency of θ ($\forall y \in \mathrm{Vars}(U), y\theta\theta = y\theta$), we get $\forall y \in \mathrm{Vars}(\mathrm{img}(\theta)), y\theta = y$. As σ is evidently a unifier of U, there exists a substitution τ such that $\sigma = \theta\tau$ Therefore, $y\sigma = y\theta\tau = y\tau$, i.e. $y\sigma = y\tau$ for all $y \in \mathrm{Vars}(\mathrm{img}(\theta))$. Thus, for any $x \in \mathrm{Vars}(T)$, $x\theta\sigma = x\theta\tau = x\sigma$. Consequently, for any term t we have $t\sigma = t\theta\sigma$. □

The following lemma is important for the complexity analysis as it shows that guessing equalities between subterms and applying the mgu of the obtained unification system actually reduces the number of distinct subterms, and thus the DAG size of the constraint system

Lemma 7. Let U be a unification system and $\theta = \text{mgu}(U)$ an idempotent most general unifier with $\text{Vars}(\text{img}(\theta)) \subseteq \text{Vars}(U)$. Then for all $t \in \text{Sub}(\text{img}(\theta))$ there exists $u \in \text{Sub}(U)$ such that $t = u\theta$.

Proof. Assume there exists terms in $\text{Sub}(\text{img}(\theta))$ that are not equal to the instance of a term in $\text{Sub}(U)$. Since this set is finite, let t be maximal among these terms for the subterm relation. If t is also maximal in $\text{Sub}(\text{img}(\theta))$, then there exits $x \in \text{Vars}(U)$ such that $x\theta = t$, a contradiction. Thus there exists a function symbol f and a term $t' = f(\dots, t, \dots)$ in $\text{Sub}(\text{img}(\theta))$. By the maximality of t there exists $u' \in \text{Sub}(U)$ such that $u'\theta = t$. If u' is not a variable, the syntactic equality implies that $u' = f(\dots, u, \dots)$ with $u\theta = t$, again a contradiction. Thus there exists a variable u' in $\text{Sub}(U)$ such that $u'\theta = f(\dots, t, \dots) = t'$. The standard procedure computing a solved form U' of a syntactic unification system U is such that $\text{Sub}(U') \subseteq \text{Sub}(U)$ (before variable identification), and the fact that $u'\theta$ is not a variable implies that U' contains an equation $u'' =_? t''$ with $u''\theta = u'\theta$ (the variables are identified in a subsequent step) and $t'' \in (\text{Sub}(U') \setminus \text{Vars}(U')) \subseteq (\text{Sub}(U) \setminus \text{Vars}(U))$. Thus there exists $t'' \in (\text{Sub}(U) \setminus \text{Vars}(U))$ such that $t''\theta = t'$. Since t'' is not a variable, it contains a subterm u (also in $\text{Sub}(U)$) such that $u\theta = t$. □

Lemma 8. Let $\mathcal{S}$ be a constraint system, σ be a ground substitution such that $\sigma \models \mathcal{S}$.

Then there exists a set of terms T, a substitution τ of domain $\text{Var}(T) \setminus \text{Var}(\mathcal{S})$, a substitution θ and a $(\mathcal{S}\theta, \sigma)$-compliant derivation D such that

- D is $(T, \sigma \cup \tau)$-maximal and one-to-one localized by T for $\sigma \cup \tau$
- $\sigma \cup \tau = \theta(\sigma \cup \tau)$
- $\text{Sub}(\mathcal{S}\theta) \subseteq T$
- T and θ are of size bounded by $u \times v \times |\text{Sub}(\mathcal{S})|$ with u a fixed coefficient depending on how data are represented, and v the size of the deduction system.

Proof. Under the same assumptions, by Lemma 5, there exists $T_0 \supseteq \text{Sub}(\mathcal{S})$ of size bounded by $v \times |\text{Sub}(\mathcal{S})|$, with v the size of the deduction system, and τ of domain $\text{Var}(T_0) \setminus \text{Var}(\mathcal{S})$ such that there exists a $(T_0, \sigma\cup\tau)$-maximal and $(\mathcal{S}, \sigma)$-compliant derivation D which is localized by T_0 for the same substitution $\sigma' = \sigma \cup \tau$.

Let $\mathcal{U} = \{t =_? t' : t, t' \in \text{Sub}(T_0) \text{ and } t\sigma' = t'\sigma'\}$. The unification system $\mathcal{U}$ has a unifier σ' and thus has a most general solution θ and by Lemma 6 we assume $\sigma' = \theta\sigma'$. Let $T = \text{Sub}(T_0)\theta$.

Since $\text{Sub}(\mathcal{S}) \subseteq T_0$ we have $\text{Sub}(\mathcal{S}\theta) \subseteq \text{Sub}(T_0\theta)$. Since θ is a most general unifier of $\mathcal{U}$ and $\text{Sub}(\mathcal{U}) = \text{Sub}(T_0)$ we have $\text{Sub}(T_0\theta) = \text{Sub}(T_0)\theta$ by Lemma 7. This implies *(i)* $\text{Sub}(\mathcal{S}\theta) \subseteq T$, *(ii)* θ is of size bounded by $u_1 \times |\text{Sub}(T_0)|$ and thus by $u_1 \times v \times |\text{Sub}(\mathcal{S})|$, with u_1 a fixed coefficient depending on how substitutions are represented, and *(iii)* T is of size bounded by $u \times v \times |\text{Sub}(\mathcal{S})|$, with $u > u_1$ a similar coefficient for substitutions and sets. The exact value is left to the reader. Moreover, as $\sigma' = \theta\sigma'$ we have $\text{Sub}(T)\sigma' = \text{Sub}(T_0)\sigma'$ and thus from D is (T_0, σ')-maximal follows D is (T, σ')-maximal.

Finally let us prove that D is one-to-one localized by T. By contradiction assume there exists $t, t' \in \text{Sub}(T)$ such that $t\sigma' = t'\sigma'$ but $t \neq t'$. Since $T = \text{Sub}(T_0\theta)$ there exists $t_0, t'_0 \in \text{Sub}(T_0)$ such that $t_0\theta \neq t'_0\theta$ but $t_0\theta\sigma' = t'_0\theta\sigma'$. This contradicts either the definition of $\mathcal{U}$ or the definition of θ as one of its unifier.

From D is $(\mathcal{S}, \sigma)$-compliant and $\sigma = \theta\sigma$ we have D is $(\mathcal{S}\theta, \sigma)$-compliant. □

4.4. Milestone sequence

In addition to retrace the deduction steps performed in D we want to track which terms relevant to $\mathcal{S}$ are deduced in T, and in which order.

Definition 4.3 (Milestone sequence)**.** A *milestone sequence* $\vec{T}$ is a finite sequence of annotated terms $?t$ or $\rightarrow t$. Let T be a set of terms, $D = (l_i \rightarrow r_i)_{1 \leq i \leq m}$ be a derivation, and σ be a ground substitution injective on $\mathrm{Sub}(T)$. A milestone sequence $\vec{T}[1 : n]$ is the (T, σ)-milestone sequence of D if there exists a strictly increasing function $\alpha : \{1, \ldots, n\} \rightarrow \{1, \ldots, m\}$ such that for every $1 \leq i \leq n$ we have:

(1) if $\vec{T}[i] = ?t$ then $D[\alpha(i)] = ?t\sigma$, and conversely if $D[i] = ?t\sigma$ then $j \in \mathrm{img}(\alpha)$;
(2) if $\vec{T}[i] = \rightarrow t$ then $D[\alpha(i)] = l_i \rightarrow t\sigma$ is a standard deduction rule; Conversely if there exists $t \in \mathrm{Sub}(T)$ such that $D[i] = l_i \rightarrow t\sigma$ then $i \in \mathrm{img}(\alpha)$

We are now examining necessary and sufficient conditions on a sequence of terms to be a good milestone sequence of a derivation. Intuitively, we want to have guessed a feasible ordering on the terms, and thus the first step towards such a definition consists in defining places for terms in a milestone sequence.

Definition 4.4 (Positions in a milestone sequence)**.** The *position* of a term t in a milestone sequence $\vec{T}$, denoted $\mathrm{Pos}_T(t)$ is either:

- ∞ if t does not occur in T;
- or, if i is minimal such that $\vec{T}[i] = \twoheadrightarrow t$, the number of non-standard deductions in $\vec{T}[1 : i]$.

Given a term $t \notin \mathcal{X}$, we let Ω_t be the set of instances of deduction rules whose right-hand side is t.

Definition 4.5 (Indices in a milestone sequence)**.** The *indice* of a (non-ground) instance of a standard deduction rule $t_1, \ldots, t_n \rightarrow t$ is denoted $\mathrm{Ind}_T(t_1, \ldots, t_n \rightarrow t)$ and is equal to $\max_{1 \leq i \leq n}(\mathrm{Pos}_T(t_n))$. The indice is extended to non-standard deductions by setting $\mathrm{Ind}_T(?t) = \mathrm{Pos}_T(t)$.

We can now define *proper* milestone sequences with regard to a set of terms T, *i.e.* sequences of terms that are candidates for being the milestone sequence of a $(\mathrm{Sub}(T), \sigma)$-maximal derivation. The two rules that have to be obeyed are *(i)* a term appearing as deduced in the sequence must be deducible and *(ii)* this deduction must be performed before the reception of any message. We express these conditions by comparing the position of a term t with the indices of rules in Ω_t. We add two other properties that will permit to have a tight relationship between proper milestone sequences and maximal derivations.

Definition 4.6 (Proper milestone sequence)**.** Let T be a set of terms and $\vec{T}$ be a sequence of $?t$ or $\rightarrow t$, for $t \in T$. We say that $\vec{T}$ is *proper* with regard to T if:

(1) For every $t \in \mathrm{Sub}(T)$, $\min_{d \in \Omega_t}(\mathrm{Ind}_T(d)) = \mathrm{Pos}_T(t)$;
(2) For every indice i in $\vec{T}$, $x \in \mathrm{Vars}\left(\vec{T}[i]\right)$ implies there exists $j \leq x$ such that $\vec{T}[j] = \rightarrow x$;

(3) For every indice i such that $\vec{T}[i] =?t$ and $\not\twoheadrightarrow t \in \vec{T}[1:i-1]$ either $i = |\vec{T}|$ or $\vec{T}[i+1] =?t'$.

5. Deciding constraint systems

Instead of trying to find a solution σ of a constraint system $\mathcal{S}$, we focus on its satisfiability by giving a necessary and sufficient condition in terms of existence of a proper milestone sequence within certain bounds. Lemma 9 states that if a (T,σ) maximal derivation D is one-to-one localized by T for σ then the (T,σ) milestone sequence of D is proper. In Theorem 1, this lemma is employed to prove that if a constraint system is satisfiable then there is a proper milestone sequence that can be connected to $\mathcal{S}$. Thus guessing T, θ, and $\vec{T}$, all of size polynomially bounded *wrt* the size of the deduction system and of the constraint system and checking the properties of $\vec{T}$ is *complete*. Then we prove in Lemma 10 that this algorithm is *sound*, *i.e.* that if a T, θ and $\vec{T}$ satisfy all the checks performed then there exists a $(\mathcal{S},\sigma)$-compliant derivation D which is (T,σ)-maximal and a proof of $\sigma \models \mathcal{S}$. Finally, the termination of this algorithm is trivial since every check is performed in time linear in the size of the constraint system and of the deduction system. Thus the satisfiability of constraint systems is decidable in non-deterministic polynomial time.

First let us proceed with the lemma essential to prove the completeness of our algorithm by constructing a proper milestone sequence from a maximal derivation.

Lemma 9. Let T be a set of terms, σ be a substitution, and D be a (T,σ)-maximal derivation which is one-to-one localized by T for σ. Then the (T,σ) milestone sequence of D is a proper milestone sequence with regard to T.

Proof. Given the maximal derivation D and since T one-to-one localizes D for σ, we construct the milestone sequence $\vec{T}$ as follows. Let n be the sum of the number of non-standard deductions in D and of the number of terms in $\mathrm{Sub}(T)\sigma$ deduced by a standard rule in D. We construct the partial strictly increasing surjective function $\alpha : \{1,\ldots,m\} \to \{1,\ldots,n\}$ and the milestone sequence $\vec{T}$ as follows. The domain of α is the set of indices i such that either $D[i]$ is a non-standard deduction rule or a standard deduction rule $l \to r$ with $r \in \mathrm{Sub}(T)\sigma$. Since this domain is totally ordered and of size n, and α is strictly increasing into $\{1,\ldots,n\}$, it is uniquely defined and onto. For $i \in \mathrm{Dom}(\beta)$ we define:

- If $D[i]$ is a standard deduction $l \to r\sigma$ with $r \in \mathrm{Sub}(T)$, we set $\vec{T}[\alpha(i)] =\!\!\to r$;
- If $D[i]$ is a non-standard deduction rule $?r\sigma$ with $r \in \mathrm{Sub}(T)$, we set $\vec{T}[\alpha(i)] =?r$

By construction, and using the (into) function α^{-1}, $\vec{T}$ is the (T,σ)-milestone sequence of D.

Thus we just have to prove that $\vec{T}$ is a proper milestone sequence. Let N be the number of non-standard rules in D, and $i_1,\ldots,i_N$ be the indices of these rules in D, and $k_{i_1}\sigma,\ldots,k_{i_N}\sigma$ (with $k_{i_j} \in \mathrm{Sub}(T)$ for $j = 1,\ldots,N$) be the messages received by these rules. Note that by construction all the indices $i_1,\ldots,i_M$ are in the domain of α.

Let us prove the different points independently.

Claim 1. *For every* $t \in \mathrm{Sub}(T)$, $\min_{d\in\Omega_t}(\mathrm{Ind}_T(d)) = \mathrm{Pos}_T(t)$.

Proof of the claim. Since D is (T,σ)-maximal, for every $t \in \mathrm{Sub}(T)$, $t\sigma \in \mathrm{Der}(\mathrm{R}_D(i))$ (for some $i \leq |D|$) if, and only if, $t\sigma \in \mathrm{R}_D(\mathrm{Next}_D(i)-1)$. For $t \in \mathrm{Sub}(T)$ let j_t be minimal such that $t\sigma \in \mathrm{Der}(\mathrm{R}_D(j_t))$ (with $j_t = \infty$ if $t\sigma \notin \mathrm{Der}(\mathrm{R}_D(|D|))$), and let M_t be maximal such that $i_{M_t} \leq j_t$. By construction every $j_t < \infty$ is in the domain of α, and since the latter is strictly increasing we have in this case:

$$\alpha(i_{M_t}) \leq \alpha(j_t) < \alpha(i_{M_t+1})$$

whereas if $j_t = \infty$ the term $t\sigma$ is not deduced in D, and thus by construction is not occurring labelled in $\vec{T}$.

Since D is one-to-one localized by T for the substitution σ, for each standard deduction $l \to t\sigma$ with $t \in \mathrm{Sub}(T) \setminus \mathcal{X}$ there exists $t_1,\ldots,t_n \in \mathrm{Sub}(T)$ such that $t_1\sigma,\ldots,t_n\sigma$ are deduced before $t\sigma$ in D and $t_1,\ldots,t_n \to t$ is an instance of a standard deduction rule. Since α is increasing, by construction the terms $t_1,\ldots,t_n$ occur before t in $\vec{T}$. Thus for all $t \in \mathrm{Sub}(T)$ we have $\mathrm{Pos}_T(t) \geq \mathrm{Ind}_T(t)$.

Assume the subset $H \subseteq \mathrm{Sub}(T)$ of terms t such that $\mathrm{Pos}_T(t) > \mathrm{Ind}_T(t)$ is not empty. If this set contains a term of finite indice, let $t \in H$ be such that i_t is minimal among the terms in H. Since $t \in H$, there exists an instance $t_1,\ldots,t_n \to t$ of a deduction rule such that $\max_{1\leq k\leq n}(\mathrm{Pos}_T(t_k)) < \mathrm{Pos}_T(t)$, and thus for $1 \leq k \leq n$ we have $\alpha(i_{t_k}) < \alpha(i_{M_t})$. Since α is increasing and into, for $1 \leq k \leq n$, we have $i_{t_k} < i_{M_t}$. Since $t_1\sigma,\ldots,t_n\sigma \to t\sigma$ is a ground instance of a deduction rule, this contradicts that the minimal i such that $t\sigma \in \mathrm{Der}(\mathrm{R}_D(i))$ is greater than or equal to i_M, and thus this contradicts the maximality of D. Otherwise, if the H contains only terms of infinite positions, there exists an instance $t_1,\ldots,t_n \to t$ of a deduction rule such that the maximum of $\mathrm{Pos}_T(t_k)$ is finite, and thus $t_1,\ldots,t_k$ occur in $\vec{T}$, but $\mathrm{Pos}_T(t)$ is infinite, *i.e.* t does not occur in $\vec{T}$. By construction of $\vec{T}$ this means that $t_1\sigma,\ldots,t_n\sigma \in \mathrm{R}_D(|D|)$, so since $t_1\sigma,\ldots,t_n\sigma \to t_\sigma$ is a ground instance of a standard deduction rule we must have $t\sigma \in \mathrm{Der}(\mathrm{R}_D(|D|))$. Since D is maximal and $t\sigma \notin \mathrm{R}_D(|D|)$, we must also have $t\sigma \notin \mathrm{Der}(\mathrm{R}_D(|D|))$, a contradiction. Hence in all cases H must be empty, and thus, for all $t \in \mathrm{Sub}(T)$ we have $\mathrm{Pos}_T(t) = \mathrm{Ind}_T(t)$, and $\vec{T}$ is a proper milestone sequence. ◇

Claim 2. *For every indice i in $\vec{T}$, $x \in \mathrm{Vars}\left(\vec{T}[i]\right)$ implies there exists $j \leq x$ such that $\vec{T}[j] =\!\!\to x$.*

Proof of the claim. If $x \in \mathrm{Vars}\left(\vec{T}[i]\right)$ then there exists corresponding deduction $D[j]$ that deduces term $\vec{T}[i]\sigma$. Then by Lemma 4 there exists $k < j$ such that $D[j]$ deduces by a standard rule $x\sigma'$. From the injectivity of σ follows that x is the only term of $\mathrm{Sub}(T)$ having σ image equal $x\sigma$. Thus, by definition of milestone sequence, there exists $m < i$ such that $\vec{T}[m] =\!\!\to x$. ◇

Claim 3. *For every indice i such that $\vec{T}[i] =?t$ and $\mathrm{Pos}_T(t) < i$ either $i = |\vec{T}|$ or $\vec{T}[i+1] =?t'$.*

Proof of the claim. Assume that $i < |\vec{T}|$, *i.e.* that $\vec{T}[i+1]$ is defined. If $\vec{T}[i+1] =\!\!\to s$, then by construction we have $D[\alpha^{-1}(i+1)] = l \to s\sigma$, and thus $s\sigma \in$

$(\mathrm{Der}(\mathrm{R}_D(i)) \setminus \mathrm{Der}(\mathrm{R}_D(i-1)))$. This contradicts $\mathrm{Pos}_T(t) < i$, as the latter implies $t\sigma \in \mathrm{Der}(\mathrm{R}_D(i-1))$ and thus $\mathrm{Der}(\mathrm{R}_D(i)) \setminus \mathrm{Der}(\mathrm{R}_D(i-1)) = \emptyset$ ◇

□

Conversely, a maximal derivation can be build once given a proper milestone sequence.

Lemma 10. If $\vec{T}$ is a proper milestone sequence for a set of terms T, and there is a substitution σ and a derivation D such that $\vec{T}$ is the milestone sequence of D, then T one-to-one localizes D for σ, and D is (T, σ)-maximal.

Proof. For each $t \in \mathrm{Sub}(T) \setminus \mathcal{X}$ such that there exists i with $\vec{T}[i] =\!\!\rightarrow t$, let l_t be a set of terms such that, for each $s \in l_t$, there exists $j < i$ with $\vec{T}[j] =\!\!\rightarrow s$ or $\vec{T}[j] =?s$. Assume $\vec{T}$ contains N variables, *i.e.* modulo renamming $\mathrm{Sub}(T) \cap \mathcal{X} = \{x_1, \ldots, x_N\}$. For $x \in \mathrm{Vars}(T)$ we let $c_x \in \mathcal{C}_{\mathrm{med}}$ be a constant not occurring elsewhere, and t_x be either:

- If $x \in \mathrm{Vars}\left(\vec{T}\right)$: the last received message before the deduction of x, or the nonce c_0 if no such term exists;
- Otherwise, a constant `Secret` no occurring in $\mathrm{Sub}(T)$ nor in the rules of the deduction system (and thus not in $\mathcal{C}_{\mathrm{med}}$)

We let $\sigma = \{x \mapsto f(c_x, t_x\sigma)\}_{x \in \mathrm{Sub}(T) \cap \mathcal{X}}$. Note this substitution is well defined by the second property of proper milestone sequences. We define the sequence of deductions D as follows:

$$D =\!\!\rightarrow c_0, \rightarrow c_{x_1}, \ldots, \rightarrow c_{x_N}, D_{\vec{T}}$$

where the sequence $D_{\vec{T}}$ of deduction rules is defined by:

$$D_{\vec{T}}[i] = \begin{cases} l_t\sigma \rightarrow t\sigma & \text{if } \vec{T}[i] =\!\!\rightarrow t \notin \mathcal{X} \\ c_x, t_x\sigma \rightarrow x\sigma & \text{if } \vec{T}[i] =\!\!\rightarrow x \in \mathcal{X} \\ ?t\sigma & \text{if } \vec{T}[i] =?t \end{cases}$$

Given the constraint on f, σ maps variables to distinct ground values that cannot occur in $(\mathrm{Sub}(T) \setminus \mathcal{X})\sigma$. Thus, by construction, D is one-to-one localized by T for σ. Let us prove that D is (T, σ) maximal. To prove this, let t be an arbitrary term in $\mathrm{Sub}(T)$, and let i be an indice in D.

- If $t\sigma \in \mathrm{R}_D(\mathrm{Next}_D(i) - 1)$ then since D is a derivation and $\mathrm{Next}_D(i) - 1 \geq i$, we have $t\sigma \in \mathrm{Der}(\mathrm{R}_D(i))$;
- Conversely, assume $t\sigma \in \mathrm{Der}(\mathrm{R}_D(i))$. Let us prove that $t\sigma \in \mathrm{R}_D(\mathrm{Next}_D(i) - 1)$. By contradiction and wlog assume that i is minimal such that there exists $t \in \mathrm{Sub}(T)$ such that $t\sigma \in \mathrm{Der}(\mathrm{R}_D(i))$ but $t\sigma \notin \mathrm{R}_D(\mathrm{Next}_D(i) - 1)$.
  - If $t \in \mathrm{Vars}\left(\vec{T}\right)$, then by the second property of proper milestone sequences there exists j with $\vec{T}[j] = t$, j_t maximal such that $\vec{T}[j_t] =?s$ and $j_t \leq j$, and by construction $t\sigma = f(c_t, s\sigma)$. Let $D[\alpha^{-1}(j_t)] =?s\sigma$. Given the third property of proper milestone sequences, $\not\twoheadrightarrow s$ does not occur in $\vec{T}[1 : \alpha(i) - 1]$. Thus by minimality of i we have $\not\twoheadrightarrow s \notin \vec{T}[1 : j_t - 1]$ implies $s\sigma \notin \mathrm{Der}(\mathrm{R}_D(i-1))$. By construction if $\alpha(i) \geq j_t$ then in D we have $t\sigma \in \mathrm{R}_D(\mathrm{Next}_D(i) - 1)$. If $\alpha(i) < j_t$, since no non-variable subterm of T is equal to $t\sigma$, by Lemma 4 the last deduction in a minimal derivation deducing $t\sigma$ must be a composition, and thus $s\sigma \in \mathrm{Der}(\mathrm{R}_D(i-1))$, a contradiction.

- If $t \in \mathrm{Vars}\,(T) \setminus \mathrm{Vars}\left(\vec{T}\right)$, since by construction there is no subterm of D that contains the constant $\mathtt{secret} \notin \mathcal{C}_{\mathrm{med}}$, $t\sigma$ cannot be deduced from $\mathrm{R}_D\,(|D|)$: by contradiction, one would consider the minimal indice m in a derivation deducing $t\sigma$ such that $\mathtt{secret}$ occurs in the right-hand side of a rule. This would have to be a composition, which is impossible.
- If $t \in \mathrm{Sub}(T) \setminus \mathcal{X}$, we note that by definition of σ and since f does not occur in any decomposition rule, we have $l\sigma \to t\sigma$ if, and only if, $l \to t$. Since $\vec{T}$ is proper, we have $\mathrm{Ind}_T\,(t) = \mathrm{Pos}_T\,(t)$, and thus every deduction $l\sigma \to t\sigma$ is such that at least one term $s \in l$ is deduced in D at a step after i

□

We are now able to prove our main theorem which provides a sound and complete criterion for deciding whether a constraint system is satisfiable.

Theorem 1. A constraint system $\mathcal{S}$ is satisfiable if, and only if, there exists a set of terms T and a substitution θ *both* of sizes bounded by $P(v, |\mathrm{Sub}(\mathcal{S})|)$ with P some fixed polynom depending on the representation of data, such that $\mathrm{Sub}(\mathcal{S})\theta \subseteq T$, plus a proper milestone sequence $\vec{T}$ and a strictly increasing mapping α such that:

(1) $(\mathcal{S}\theta)[i] =!t$ implies $\vec{T}[\alpha(i)] =?t$ and if $\vec{T}[i] =?t$ then i is in the image of α;

(2) $(\mathcal{S}\theta)[i] =?t$ implies $\mathrm{Pos}_T\,(t) \leq \mathrm{Pos}_T\left(\vec{T}[\alpha(\mathrm{prev}_{\mathcal{S}\theta}(i))]\right)$;

(3) $(\mathcal{S}\theta)[i] = \natural t$ implies $\mathrm{Pos}_T\,(t) > \mathrm{Pos}_T\left(\vec{T}[\alpha(\mathrm{prev}_{\mathcal{S}\theta}(i))]\right)$.

Proof. If $\mathcal{S}$ is satisfiable then by Lemma 8 there exist T, σ, τ and θ as advertised by this lemma, and a $(T, (\sigma \cup \tau))$-maximal $(\mathcal{S}\theta, \sigma)$-compliant derivation D which is a proof of $\sigma \models \mathcal{S}$ and one-to-one localized by T for $(\sigma \cup \tau)$. Thus there exists a proper milestone sequence $\vec{T}$ by Lemma 9. The fact that D is $(\mathcal{S}, \sigma)$-compliant yields the first point. The facts that it is $(T, \sigma \cup \tau)$-maximal, one-to-one localized by T for $(\sigma \cup \tau)$ and that $\mathrm{Sub}(\mathcal{S})\theta \subseteq T$ yields the second and third points. Note that the sizes of T and θ are already bounded thanks to Lemma 8.

Conversely, if there exists T, $\vec{T}$ and θ as advertised, there exists by Lemma 10 one constructs from $\vec{T}$ a substitution σ and a (T, σ)-maximal and $(\mathcal{S}, \sigma)$-compliant derivation D. The properties of proper milestone sequences ensure that the conditions of Lemma 2 are met, and thus that D is a proof of $\sigma \models \mathcal{S}$. □

From the previous result we can directly derive a trivial NP-decision procedure for constraint systems satisfiability: even if not formally computed here to avoid any dependency on the way data are represented in practice, the polynom P bounding the size of a set of terms is known and computable as soon as the representations of data (substitutions, deduction system) are chosen. Assume v is the size of the deduction system. All we have to do is to: guess a set of terms of size lower than $|\mathrm{Sub}(\mathcal{S})| \times v$, a substitution θ of size bounded by $P(v, |\mathrm{Sub}(\mathcal{S})|)$, a milestone sequence $\vec{T}$ of length less than or equal to $|\mathrm{Sub}(T)|$, and check (in polynomial time) whether $\mathrm{Sub}(\mathcal{S}) \subseteq \mathrm{Sub}(T\theta)$, whether $\vec{T}$ is a proper milestone sequence with regard to T and has all the requisite properties from Theorem 1, and whether there exists a function α mapping the terms sent in the constraint system to their θ-instance in $\vec{T}$. The NP-hardness is entailed by the NP-hardness of solving constraint systems without negative constraints (Rusinowitch and Turuani, 2003). Note that the deduction system, and thus its size v, is part of the input data for NP-completeness: the one shown for LOP in Section 2 was only an example.

6. Implementation and computer experiments

6.1. Solving negative constraints in Cl-Atse

Cl-Atse is a *Constraint Logic based Attack Searcher* (Turuani, 2006) for security protocols and services specified in HLPSL (Chevalier et al., 2004) or ASLan languages (Armando et al., 2012). It runs the protocol or set of services along all possible traces and dynamically collects the constraints over terms and messages, over security properties, etc. Therefore, Cl-Atse's engine is primarily a constraint solver, reducing them down to normal forms instead of bounding derivations. Cl-Atse only handles a special subterm theory modelling symmetric and asymmetric encryption, that we call here Dolev Yao theory. Since for our target examples Dolev Yao theory was sufficient we have preferred to adapt the existing procedure of Cl-Atse instead of directly implementing the general but highly inefficient decision procedure for negative constraints from previous section.

Let us recall the way the standard version of Cl-Atse works with positive constraints only. The tool splits a constraints system into sub-systems and reduces them, while preserving the set of solutions The main goal is to reduce sets of constraints systems such that in all the $?t$ constraints t is limited to be a variable and all the ! constraints are decomposed with regard to the new constraints that this may require. A system of this form is said reduced, and (without $\natural$ constraint) it is easily satisfiable by filling variables with fresh nonces created by the mediator at the start: all the $?X$ constraints can use them directly. This points out that once a reduced constraint (without $\natural$ constraint) system is reached, satisfiability is ensured by any assignment of the variables.

The tool has been extended with negative constraints (i.e. $\natural$) which serve as guards during the reduction. Preserving the process already working for positive constraints (i.e. ?), the tool does not reduce $\natural$ as it does for ?. Instead, it uses it to eliminate sub-systems that syntactically admit no solutions with regard to the negative constraints. Let $C_{\mathcal{X}}$ be a set of fresh nonces generated by the intruder at the start, and let $C'_{\mathcal{X}}$ be a set of fresh nonces known by *nobody* (at the start of later). Let γ be an injective substitution from $\mathcal{X}$ to $C_{\mathcal{X}} \cup C'_{\mathcal{X}}$ such that $V\gamma \subseteq C_{\mathcal{X}}$ with $V = \mathrm{Vars}\,(In(S) \cup Out(S))$, i.e. any variable shown in at least one positive constraint get a value known by the intruder, and $(\mathrm{Vars}\,(S) \backslash V)\gamma \subseteq C'_{\mathcal{X}}$ i.e. any variable only shown in the negative constraints get a value unkonwn to the intruder. We say that a constraint system S is contradictory iff there exists S_1, t and S_2 such that $S = S_1.\natural t.S_2$ and $t\gamma \in Der(In(S_1)\gamma \cup Out(S_1)\gamma)$. The tool does not compute γ explicitly but simply consider variables like nonces for this test. The idea of this test is to: i) have a fast, incomplete, test for eliminating obviously contradicting constraint systems each time a reduced constraint system is reached: thanks to the *Determination* any solution must satisfy the negative constraints at least syntactically; ii) see that on a reduced constraint system, this syntactic test is sufficient to provide at least one solution: we know that $\gamma_{|V} : V \to C_{\mathcal{X}}$ satisfies the positive constraints as would any solution filling variables with fresh nonces known only by the intruder at the beginning; but it also satisfies the negative ones by definition of a contradictory system. Note that this method to add negative constraints over an existing constraint solving technique only works if the constraint solving algorithm for positive-only constraints produces reduced systems of suitable type.

Finally, as pointed in i) this elimination test is fast: theoretically, the testing of Der modulo γ is a derivation test in the ground case, which is known to be polynomial and done by building a set of derivable subterms from which we can compose the target;

however in practice, this ability is essential for reducing positive constraints, and already exists in the tool. For example, the purpose of decomposing the ! constraints as pointed above is precisely for collecting all the derivable subterms. This makes the test straightforward and limits the speed impact on the tool. Moreover, the tool does not reduce the whole constraint system at once: instead, it builds it by successively adding new constraints and reducing the intermediate result, e.g. each time a choice is done on the next protocol step to follow. Therefore, the contradicting negative constraints are eliminated progressively, at each step. In the end, the processing of $\natural$ has not a large impact on Cl-Atse computation time.

6.2. Integration in AVANTSSAR's Orchestrator

The AVANTSSAR Orchestrator [3] (Armando et al., 2012) is a tool built over Cl-Atse for automatic orchestration of web services along with their security policies. The orchestration problem in this context can be reduced to a protocol state reachability problem, which in turn can be converted into a classical protocol insecurity problem where the adversary plays the role of a mediator. The transformation can be described as follows: The available web services are mapped in a one-to-one fashion to security protocol parties (roles); since services and roles are both represented by sequences of message receptions and sends this transformation is straightforward (see details and examples in public Deliverable 4.2 at `www.avantssar.eu`). The Client in the orchestration problem is specified too as a sequence of message receptions and sends and therefore can be translated directly to a security protocol role also called Client. This role is then extended by a last step where it sends a special token *End_execution_client* signalling that the Client has successfully finished its execution. Now, given the derived protocol parties (corresponding to the available web services and the client) we ask, whether the intruder can learn that special token issued by the Client, i.e. whether the intruder knows that token at some point of some protocol execution (insecurity goal). All the non-disclosure policies are added to the insecurity goal. Thus, the protocol insecurity problem can now be stated in form: is there a protocol execution in presence of an intruder, such that this intruder learns the token issued by the Client at the end, without being able to infer the other messages stated in the non-disclosure policies ? If such an execution exists, the actions of the intruder in this execution can be translated back to a Mediator service satisfying both the Client communication requirements and all the non-disclosure policies. The relevant execution can be found by the Cl-Atse procedure for solving negative constraints as described in Subsection 6.1.

In practice, the AVANTSSAR Orchestrator also relies on the ASLan specification language for web services created for AVANTSSAR platform, a translation tool dedicated to ASLan, and an analysis backend (possibly Cl-Atse) for validating the composed specification produced. Negative constraints have been integrated, which now supports an extended variant of the ASLan language with negative constraints of the form $not(iknows(t))$, where: t is a term; $iknows(t)$ is a constraint requiring that t is deducible by the intruder at the moment when the transition containing it is run; and $not(iknows(t))$ is the negation of this, i.e. there must not exist any sequence of intruder deduction rules capable of producing t at that time.

[3] Available online at `https://cassis.loria.fr/OrchestratorWI/`.

6.3. Experiments.

The LOP motivating example from Section 2, which is inspired by the Loan Origination case study from AVANTSSAR, has already been described in Section 2 and requires non-deductibility constraints. Therefore, it is up to the AVANTSSAR's Orchestrator to generate a non-trivial mediator that is unable to *deduce*, i.e. even through calculations and beyond simple eavesdropping, the client's private data and is unable to invalidate the security policy of any agent involved in the exchange.

In the standard AVANTSSAR framework, it was not possible to express directly non-disclosure policies nor separation-of-duty policies. This gets now possible thanks to the introduction of negative constraints in both Cl-Atse and AVANTSSAR's Orchestrator. For example, the client model in ASLan++ before this extension, i.e. without any negative constraints, is the following:

```
 1.  entity Client(Actor, Pep, M: agent, Amount: text) {
 2.    symbols
 3.      Ephemeral_key : symmetric_key;
 4.      Resp_A, Resp_B, End_execution_client : text;
 5.      A, B : agent;
 6.    body {
 7.      Actor -> M : {g(Actor).loan.Pep}_inv(pk(Actor));
 8.      M -> Actor : ?A.?B;
 9.      Ephemeral_key := fresh();
10.      Actor -> M : {Amount.Actor.Ephemeral_key}_pk(A).
11.                   {Amount.Actor.Ephemeral_key}_pk(B);
12.      M -> Actor : {h(A.Amount.Actor.?Resp_A)}_inv(pk(A)).
13.                   {h(B.Amount.Actor.?Resp_B)}_inv(pk(B)).
14.                   {|?Resp_A|}_Ephemeral_key.
15.                   {|?Resp_B|}_Ephemeral_key;
19.      secrecy_End_of_execution:(End_execution_client) := fresh();
20.      Actor -> Pep: {Amount.Actor.A.Resp_A.B.Resp_B}_pk(Pep).
21.                    {h(A.Amount.Actor.Resp_A)}_inv(pk(A)).
22.                    {h(B.Amount.Actor.Resp_B)}_inv(pk(B)).
23.                    End_execution_client;
24.  }}
```

The ASLan++ term notation matches the one in this paper (e.g. `{|M|}_K` is $\{|M|\}_K$), except that `?A` designates the new value of A in the current action (probably a reception but not only) instead of just the action that receives it. In here, the client played by agent *Actor* communicates with his bank *Pep* and the mediator M from which he receives tho clerk's names A and B (at line 8) after sending his loan examination request (at line 7). Then, he encrypts and sends his private loan data, simplified here to the amount itself, for the clerks through the mediator (at lines 10 and 11). In return, he expects signed responses protected by his fresh ephemeral key (at lines 12 to 15), which can then be used to contact the loan provider *Pep* (at lines 20 to 22). The sending of token *End_execution_client* defines the client's success (at lines 23). Once translated to ASLan using the AVANTSSAR's Aslanpp connector, this role definition can be extended with new guards:

```
16.        not(iknows(Amount));
17.        not(iknows(Resp_A));
18.        not(iknows(Resp_B));
```

ensuring that the client won't successfully terminate the run by sending *End_execution_client* if the intruder (playing the mediator M) is capable to build or deduce the loan's amount or any of the clerk's private response. Similarly, the clerk's ASLan model is extended with negative constraints encoding separation of duty with the following idea: i) Any honest agent A taking a role in the process sends a $g(A)$ token to the mediator, like e.g. $g(Actor)$ at the client's line 7; ii) Relationships between agents are modeled through encryptions known by the mediator, like e.g. $\{g(C)\}_{g(B)}$ if C is linked to B, and vice versa; iii) when a new agent C is selected to be a clerk, a negative constraint $not(g(C))$ ensures that there exists no chain of relationships of any length that could link him to some agent already involved in the exchange.

As a result, we can show the effects of negative constraints in this application by experimenting variants of the model where the mediator is, or is not, restricted by negative constraints and where clerks are, or are not, available to the mediator:

(1) Without the negative constraints, and with or without any clerk, the Orchestrator finds a mediator that assumes himself the role of a clerk, which is precisely the behavior we want to avoid since further analysis will necessary reject it. This shows that if allowed to, the mediator really has the knowledge needed to act as a clerk;

(2) With the negative constraints, but without any clerk, the Orchestrator does not find any valid orchestration. This is logical since without any external agent playing as a clerk, and with constraints like $not(iknows(Amount))$ preventing the mediator from being able to deduce the loan *Amount*, and thus, from being a clerk himself, it is impossible to satisfy the client's request.

(3) With both the negative constraints and the clerks being available, the Orchestrator finds a good orchestration, i.e. one where all the security policies of all agents are satisfied in any trace. There, as expected the mediator delegates some parts of the process to the clerks, which appears to be a needed condition for the secret data to remain secret and non-deducible by the mediator.

For example, assume that the mediator's initial knowledge contains a client name *client*, three clerks's names *alice*, *bob* and *charlie*, an entry for *bob* and *charlie* in the relational database (i.e. $\text{rel}(g(bob), g(charlie))$), and the methods to communicate with these agents, then as expected in the point (3) above the complete specification of LOP in our setting allow the tool to produce a (new) mediator entity for this form:

```
 1. entity Mediator(M : agent) {
 2.   symbols
 3.     Loan, Req1, Req2, HResp1a, HResp1b, HResp2a, HResp2b : message;
 4.     Pep : agent;
 5.   body {
 6.     M -> alice :  request.M
 7.     alice -> M :  g(alice).pk(alice)  % Cannot create g(alice) before
 8.     M   -> bob :  request.M
 9.     bob -> M   :  g( bob ).pk( bob )  % Cannot create g( bob ) before
10.     client -> M:  {g(client).?Loan.?Pep}_inv(pk(client))
11.     M -> client:  alice.bob
12.     client -> M:  ?Req1.?Req2
13.     M -> alice :  Req1
14.     alice -> M :  {?HResp1a}_inv(pk(alice)).?HResp1b  % Cannot create before
15.     M   -> bob :  Req2
16.     bob -> M   :  {?HResp2a}_inv(pk( bob )).?HResp2b  % Cannot create before
```

```
17.     M -> client:  {HResp1a}_inv(pk(alice)).HResp1b.
18.                   {HResp2a}_inv(pk(alice)).HResp2b
19.                   % Cannot create: alice's and bob's Resp, client's Amount.
20. }}
```

7. Conclusion

We have obtained the first decision procedure for deducibility constraints with negation and we have applied it to the synthesis of mediators subject to non-disclosure policies. It has been implemented as an extension of Cl-Atse (Turuani, 2006) for the Dolev-Yao deduction system. On the Loan Origination case study, the prototype generates directly the expected orchestration. Without negative constraints undesired solutions in which the mediator impersonates the clerks were found. More details, including problem specifications, can be found at `http://cassis.loria.fr/Cl-Atse`. As in (Abadi and Cortier, 2006; Baudet, 2005) our definition of subterm deduction systems can be extended to allow ground terms in right-hand sides of decomposition rules even when they are not subterms of left-hand sides and the decidability result remains valid with minor adaptation of the proof. A more challenging extension would be to consider general constraints (as in (Avanesov et al., 2011)) with negation.

References

Abadi, M., Cortier, V., 2006. Deciding knowledge in security protocols under equational theories. Theoretical Computer Science 367 (1–2), 2 – 32.

Armando, A., Arsac, W., Avanesov, T., Barletta, M., Calvi, A., Cappai, A., Carbone, R., Chevalier, Y., Compagna, L., Cuéllar, J., Erzse, G., Frau, S., Minea, M., Mödersheim, S., von Oheimb, D., Pellegrino, G., Ponta, S. E., Rocchetto, M., Rusinowitch, M., Dashti, M. T., Turuani, M., Viganò, L., 2012. The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures. In: TACAS. Vol. 7214 of Lecture Notes in Computer Science. Springer, pp. 267–282.

Armando, A., Giunchiglia, E., Maratea, M., Ponta, S. E., 2013. Modeling and reasoning about business processes under authorization constraints: A planning-based approach. In: Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling, ICAPS 2013, Rome, Italy, June 10-14, 2013. AAAI.

Armando, A., Ponta, S. E., 2014. Model checking authorization requirements in business processes. Computers & Security 40, 1–22.

Avanesov, T., Chevalier, Y., Mekki, M., Rusinowitch, M., 2012a. Web services verification and prudent implementation. In: DPM/SETOP 2011. Vol. 7122 of Lecture Notes in Computer Science. Springer, pp. 173–189.

Avanesov, T., Chevalier, Y., Rusinowitch, M., Turuani, M., 2011. Satisfiability of general intruder constraints with and without a set constructor. CoRR abs/1103.0220.

Avanesov, T., Chevalier, Y., Rusinowitch, M., Turuani, M., 2012b. Towards the orchestration of secured services under non-disclosure policies. In: Kotenko, I. V., Skormin, V. A. (Eds.), MMM-ACNS. Vol. 7531 of Lecture Notes in Computer Science. Springer, pp. 130–145.

AVANTSSAR, 2008–2010. Automated Validation of Trust and Security of Service-Oriented Architectures, AVANTSSAR project. `http://www.avantssar.eu`.

Baudet, M., 2005. Deciding security of protocols against off-line guessing attacks. In: Proceedings of CCS'05 conference. ACM, pp. 16–25.

Chevalier, Y., Compagna, L., Cuellar, J., Hankes Drielsma, P., Mantovani, J., Mödersheim, S., Vigneron, L., September 2004. A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. Vol. 180 of Automated Software Engineering. Austrian Computer Society, Austria, pp. 193–205.

Chevalier, Y., Mekki, M., Rusinowitch, M., 2008. Automatic composition of services with security policies. In: Proceedings of SERVICES I 2008. SERVICES '08. IEEE, Washington, DC, USA, p. 529–537.

Chevalier, Y., Mekki, M., Rusinowitch, M., 2012. Orchestration under security constraints. In: Proceedings of FMCO 2010. Vol. 6957 of Lecture Notes in Computer Science. Springer, pp. 23–44.

Comon-Lundh, H., Cortier, V., Zalinescu, E., 2010. Deciding security properties for cryptographic protocols. application to key cycles. ACM Trans. Comput. Log. 11 (2).

Corin, R., Etalle, S., Saptawijaya, A., May 21–24 2006. A logic for constraint-based security protocol analysis. In: IEEE Symposium on Security and Privacy (S&P), Berkeley, California, USA. IEEE Computer Society, pp. 155–168.

Costa, G., Degano, P., Martinelli, F., 2011. Secure service orchestration in open networks. Journal of Systems Architecture - Embedded Systems Design 57 (3), 231–239.

Dolev, D., Yao, A., 1983. On the security of public key protocols. Information Theory, IEEE Transactions on 29 (2), 198–208.

Ferraiolo, D., Kuhn, R., 1992. Role-based access control. In: In 15th NIST-NCSC National Computer Security Conference. pp. 554–563.

Frau, S., Dashti, M. T., 2011. Integrated specification and verification of security protocols and policies. In: 24th IEEE Computer Security Foundations Symposium, CSF 2011, Cernay-la-Ville, France, 27-29 June. pp. 18–32.

Herzig, A., Lorini, E., Hübner, J. F., Vercouter, L., 2010. A logic of trust and reputation. Logic Journal of IGPL 18 (1), 214–244.

Kähler, D., Küsters, R., Truderung, T., 2007. Infinite state AMC-model checking for cryptographic protocols. Proceedings of the Twenty-Second Annual IEEE Symposium on Logic in Computer Science (LICS 2007), 181–192.

Kourjieh, M., december 2009. Logical Analysis and Verification of Cryptographic Protocols. Thèse de doctorat, Université Paul Sabatier, Toulouse, France.

Lorini, E., Demolombe, R., 2008. Trust and norms in the context of computer security: A logical formalization. In: van der Meyden, R., van der Torre, L. (Eds.), Deontic Logic in Computer Science. Vol. 5076 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 50–64.

Lynch, C., Meadows, C., 2005. On the relative soundness of the free algebra model for public key encryption. In: Proceedings of the 2007 FCS-ARSPA Workshop. Vol. 125 of Electronic Notes in Theoretical Computer Science. pp. 43–54.

Martinelli, F., 2005. Towards an integrated formal analysis for security and trust. In: 7th IFIP WG 6.1 International Conference, FMOODS 2005, Athens, Greece, June 15-17, 2005. pp. 115–130.

McAllester, D. A., 1993. Automatic recognition of tractability in inference relations. Journal of the ACM 40, 284–303.

Millen, J., Shmatikov, V., 2001. Constraint solving for bounded-process cryptographic protocol analysis. In: Proceedings of the 8th ACM conference on Computer and Communications Security. CCS '01. ACM, New York, NY, USA, pp. 166–175.

NESSoS, 2010–2014. Network of Excellence on Engineering Secure Future Internet Software Services and Systems, NESSoS project. `http://www.nessos-project.eu`.

Rusinowitch, M., Turuani, M., 2003. Protocol Insecurity with Finite Number of Sessions and Composed Keys is NP-complete. Theoretical Computer Science 299, 451–475.

Turuani, M., 2006. The CL-Atse Protocol Analyser. In: Term Rewriting and Applications (RTA). LNCS 4098. pp. 277–286.

Viganò, L., 2012. Towards the secure provision and consumption in the internet of services. In: Trust, Privacy and Security in Digital Business - 9th International Conference, TrustBus 2012, Vienna, Austria, September 3-7, 2012. Proceedings. Vol. 7449 of Lecture Notes in Computer

Science. Springer.
Viganò, L., 2013. The SPaCIoS project: Secure provision and consumption in the internet of services. In: 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation, Luxembourg, Luxembourg, March 18-22, 2013. IEEE.